21世纪高职高专规划教材

计算机应用系列

ASP.NET动态网站设计

王淑敏 主编　马世霞 刘丹 白艳玲 副主编

清华大学出版社

21 世纪高职高专规划教材
计算机应用系列

ASP.NET 动态网站设计

王淑敏　主编
马世霞　刘　丹　白艳玲　副主编

清华大学出版社
北　京

内 容 简 介

本书通过讲解一个完整的项目“在线考试系统”的设计与实现，详细介绍了 ASP. NET 项目开发的过程，并将 ASP. NET 所有的知识和技能穿插其中。本书共分 15 个任务进行讲解，内容包括 ASP. NET 概述及运行环境的构建，C＃语言基础与实体类创建，为在线考试系统制作导航系统，三层架构与系统框架，ADO. NET 数据库操作与数据访问层类的创建，验证控件与用户登录，ASP. NET 内置对象与登录页面完善，GridView 控件、DetailsView 控件与考生信息显示，主题与母版页，DataList 控件与试题信息管理，第三方控件与试题的信息添加，Repeater 控件与前台试题显示，用户控件与网站版权，文件及文件夹操作，网站部署与定制。

本书适用于各类高等职业院校计算机技术专业，也可作为计算机培训班的教材，还可供从事网页设计的技术人员学习参考。

图书在版编目（CIP）数据

ASP. NET 动态网站设计/王淑敏主编．—北京：清华大学出版社，2010. 7
（21 世纪高职高专规划教材. 计算机应用系列）
ISBN 978-7-302-22873-8

Ⅰ. ①A…　Ⅱ. ①王…　Ⅲ. ①主页制作－程序设计－高等学校：技术学校－教材
Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字(2010)第 099195 号

责任编辑：张龙卿(sdzlq123@163. com)
责任校对：袁　芳
责任印制：

出版发行：清华大学出版社　　**地　址**：北京清华大学学研大厦 A 座
http://www. tup. com. cn　　**邮　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup. tsinghua. edu. cn
质　量　反　馈：010-62772015，zhiliang@tup. tsinghua. edu. cn
印 刷 者：
装 订 者：
经　销：全国新华书店
开　本：185×260　　**印　张**：18. 5　　**字　数**：445 千字
版　次：2010 年 7 月第 1 版　　**印　次**：2010 年 7 月第 1 次印刷
印　数：1～0000
定　价：0. 00 元

产品编号：034787-01



前　　言

由于微软软件开发工具产品的易学易用，ASP.NET 在当今动态网页与网站设计技术中占有主导地位。根据教育部有关教育教学改革精神，结合当前高职高专教育的特点，编者在总结多年教学经验的基础上，编写了本书。

本书以一个完整的项目“在线考试系统”为导向，以任务为基本单元，将知识和技能穿插其中，讲解了 ASP.NET 技术。在确定开发项目时一方面考虑实用性；另一方面考虑到理论知识的系统性，并采用大家感兴趣、熟悉且可以操作的实际项目。

本书具有以下特点。

(1) 采用项目驱动式教材编写体系。本书将一个大的项目案例的实现划分为若干任务，分别讲解，在完成每个具体任务时，会将相关的知识点进行透彻讲解。

(2) 内容新颖，注重应用。尽管现在市场上 ASP.NET 方面的书已经有不少，但能满足教学需要的却很难找。针对这种状况，我们在编写时特别注意以下三点：一是注意知识点和工具应用的紧密结合；二是注意知识体系的更新，整个项目采用目前流行的三层架构；三是注意实用性，不但采用真实项目，代码编写也比较规范，便于学生将来快速适应工作环境。

(3) 理论知识系统性比较强。本书大部分任务都包括知识准备和任务实施，任务实施注重动手能力的培养，并逐步完成在线考试系统的制作；知识准备部分为当前任务的实施奠定了理论基础，并保证学习过程中理论知识讲解的系统性。

本书由王淑敏任主编，马世霞、刘丹、白艳玲任副主编。具体编写工作分工如下：任务 1、任务 12 由王淑敏编写，任务 3、任务 4、任务 14 由马世霞编写，任务 6、任务 8、任务 15 由刘丹编写，任务 9、任务 10 由白艳玲编写，任务 5、任务 7 由郜伟民编写，任务 11、任务 13 由胡海鹤编写，绪论部分的项目介绍、任务 2 由白林如编写。在本书的编写过程中，得到了清华大学出版社的大力支持，在此表示衷心的感谢。

虽然我们在编写过程中倾注了大量心血，但书中难免有疏漏之处，恳请广大读者批评指正。

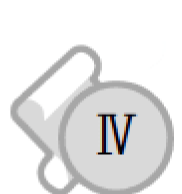
编　者

2010 年 5 月



目　　录















绪论　案例说明——在线考试系统介绍

本部分首先介绍一下本书中后面内容使用到的案例——在线考试系统，以便大家对案例的整体结构有所了解。

0.1　系统分析和开发环境

在线考试系统是学校针对学生的考试系统。学校可以通过该考试系统建立自己的网上考场，使学生直接进行网上考试，既减少了成本，又避免了笔试的烦琐过程，使考试过程变得轻松、方便；同时还可以有效地控制考试的作弊现象，确保考试的公开、公平、公正性。

该在线考试系统采用三层设计模式，分别为用户界面表示层、业务逻辑层、数据访问层，另外还有一个模型层，如图 0-1 所示。

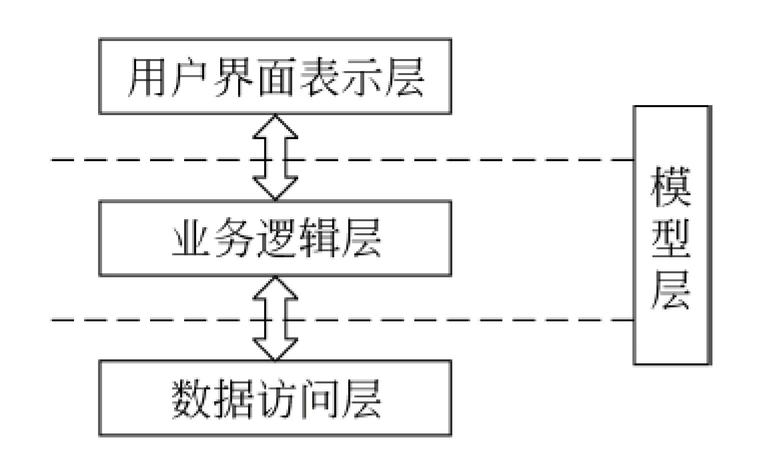


图 0-1　三层开发模式及模型层

本系统开发工具：Visual Studio 2005 和 SQL Server 2005。

0.2　系统功能模块设计

系统开发的总体任务是实现信息关系的系统化、规范化和自动化。

系统的功能模块如图 0-2 所示。

系统采用 SQL Server 2005 数据库，包含系统管理员表、题库表、科目表、试卷表、考生试卷表、试题类型表、班级表、学生表、成绩表等十余个表，如图 0-3～图 0-12 所示。



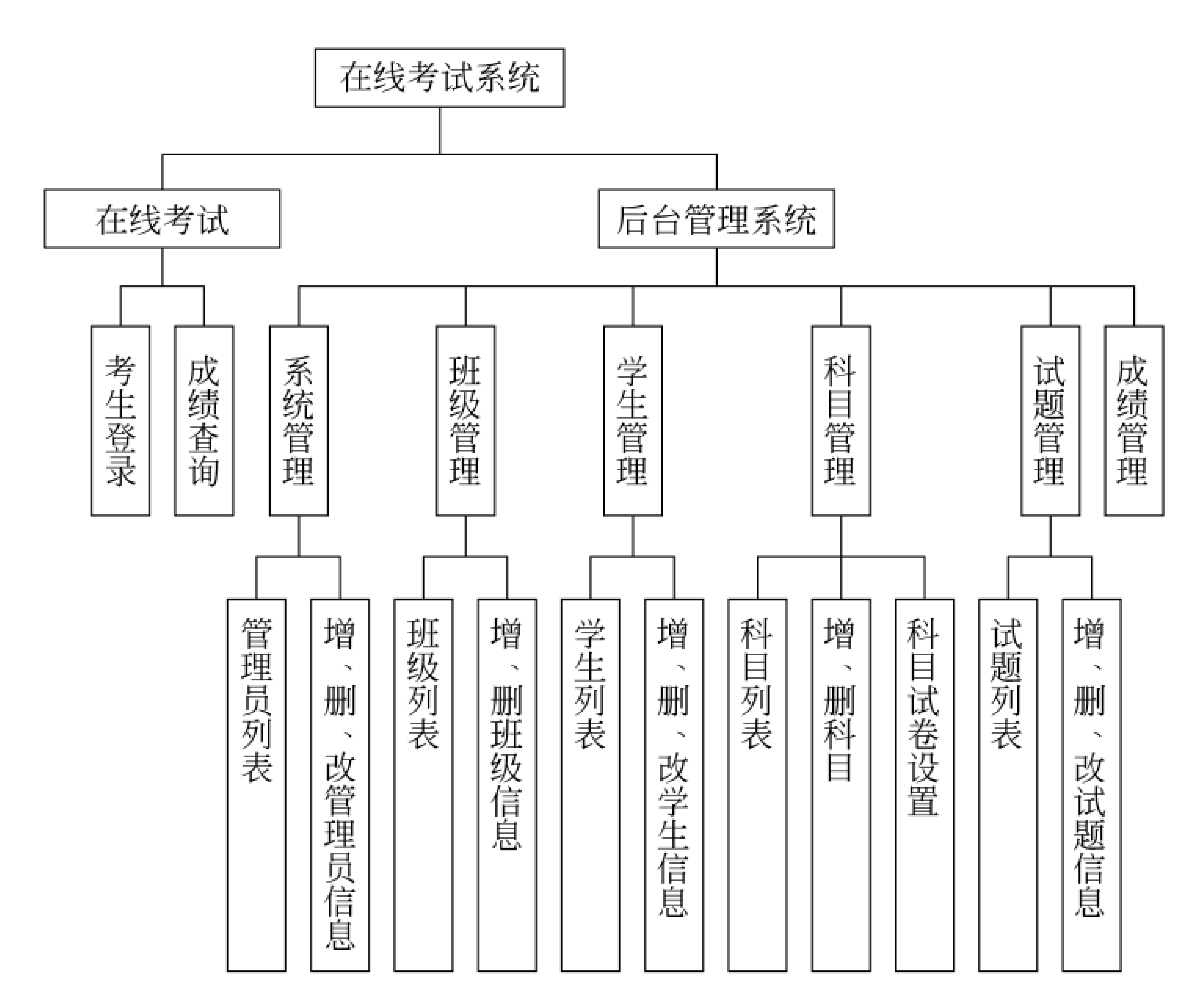


图 0-2　系统功能模块

表 - dbo.exam_admin

列名	数据类型	允许空
admin_id	int	☐
admin_name	varchar(20)	☐
admin_pass	varchar(20)	☐
mgquestions	int	☑
mgStudents	int	☑
mgSystem	int	☑

图 0-3　系统管理员表

表 - dbo.exam_allQuestions

列名	数据类型	允许空
question_id	int	☐
subject_id	int	☐
question_type	int	☐
question_subject	varchar(200)	☐
question_keys	varchar(10)	☐
a	varchar(100)	☑
b	varchar(100)	☑
c	varchar(100)	☑
d	varchar(100)	☑

图 0-4　题库表

表 - dbo.exam_allsubject

列名	数据类型	允许空
subject_id	int	☐
subject_name	varchar(50)	☐

图 0-5　科目表



表 - dbo.exam_canTry

列名	数据类型	允许空
stu_id	varchar(20)	☐
paperdb_name	varchar(50)	☑

图 0-6　试卷表(一)

表 - dbo.exam_papermg

列名	数据类型	允许空
paper_id	int	☐
subject_id	int	☐
paper_name	varchar(50)	☐
test_time	int	☐
type1_num	int	☐
type2_num	int	☐
type3_num	int	☐
score1	int	☑
score2	int	☑
score3	int	☑

图 0-7　试卷表(二)

表 - dbo.exam_papers

列名	数据类型	允许空
question_id	int	☐
subject_id	int	☐
question_type	int	☐
question_subject	varchar(200)	☐
question_key	varchar(10)	☐
a	varchar(100)	☑
b	varchar(100)	☑
c	varchar(100)	☑
d	varchar(100)	☑
user_key	varchar(50)	☑

图 0-8　考生试卷表

表 - dbo.exam_questionType

列名	数据类型	允许空
type_id	int	☐
type_name	char(10)	☐

图 0-9　试题类型表

表 - dbo.exam_stuClass

列名	数据类型	允许空
class_id	int	☐
class_name	varchar(50)	☐

图 0-10　班级表

表 - dbo.exam_students

列名	数据类型	允许空
stu_id	varchar(10)	☐
stu_name	varchar(20)	☐
class_id	int	☐
isLogin	int	☐
isSubmit	int	☐
stu_pwd	varchar(20)	☐

图 0-11　学生表



表 - dbo.exam_stuScore

列名	数据类型	允许空
score_id	int	☐
stu_id	varchar(10)	☐
subject_id	int	☐
paperdb_name	varchar(50)	☐
try_date	varchar(50)	☑
stu_score	int	☐

图 0-12　成绩表

0.3　系统运行界面

在线考试系统的操作流程如下：

首先，进入在线考试系统首页面，如图 0-13 所示。管理员可以选择“进入后台管理”选项对系统进行维护操作，如图 0-14 和图 0-15 所示。考生选择“开始考试”选项后，进入考试登录页面，如图 0-16 所示。考生选择考试科目并输入学号和密码，单击“开始考试”按钮，浏览器转到考试页面，考生可在线答题，如图 0-17 所示。



图 0-13　在线考试系统首页

欢迎登录在线考试系统

用户名：

密码：

进入系统后台　重　置

图 0-14　后台管理登录页面

后台管理

- 管理员控制面板
  - 管理员信息
    - 管理员列表
    - 添加管理员
  - 班级管理
    - 班级列表
    - 添加班级
  - 学生管理
    - 学生列表
    - 添加学生
  - 科目管理
  - 试题管理
  - 成绩管理
  - 退出

您的位置：首页 > 管理员后台 > 学生管理 > 学生列表

学号	学生姓名	班级	密码	详细	删除
0000001	白嘉辉	07级1班	0	详细	删除
0000002	武超帅	07级1班	0	详细	删除
0000003	杨光宇	07级1班	0	详细	删除
0000004	张西良	07级2班	3	详细	删除
0000005	朱彬彬	07级1班	0	详细	删除
0000006	朱志伟	07级1班	0	详细	删除
0000007	赵磊	07级1班	0	详细	删除
0000008	戴亮	07级1班	0	详细	删除
0000009	张鹏飞	07级2班	0	详细	删除
0000010	邢霖	07级1班	0	详细	删除

1 2 3 4

图 0-15　后台管理

欢迎登录在线考试系统

请选择考试科目：<-- 请选择 -->

学号：

密码：

开始考试　重　置　查看已考科目成绩

图 0-16　学生登录

《计算机基础》考试试卷

你是第8位登录的考生

一、单项选择题

1. 微型计算机（PC机）的文件系统采用的是_____结构。
   - ○环型
   - ○星型
   - ○树型
   - ○网型
2. 为解决某一特定问题而设计的指令序列称为_____。
   - ○文档
   - ○语言
   - ○程序
   - ○系统
3. 计算机病毒是一种_____。
   - ○特殊的计算机部件

图 0-17　考试页面

答题完毕后，单击“提交”按钮，系统计算考试成绩后，浏览器将被定向到成绩显示页面，如图 0-18 所示。在图 0-16 所示页面中，输入学号和密码后单击“查看已考科目成绩”按钮，也可



进入成绩显示页面(图 0-18),显示该考生的已考科目成绩。



图 0-18　成绩显示

0.4　Web.config 文件和数据库操作公共类

0.4.1　Web.config 文件

为了提高程序的灵活性和可维护性,我们把数据库连接信息存放在 Web.config 文件中。在用户界面表示层的根目录下打开 Web.config 文件,在<configuration>配置代码块中添加<connectionStrings>代码块内容,如下所示。

```
<configuration>
    <appSettings/>
    <connectionStrings>
        <add name="onLineExam" connectionString="Data Source=.;Initial Catalog=
          onLineExam1;User ID=sa;password=sa"/>
    </connectionStrings>
    <system.Web>
        ...
        <authentication mode="Forms">
            <forms name="Login" loginUrl="~/admin/AdminLogin.aspx" timeout="60"/>
        </authentication>

    </system.Web>
</configuration>
```

0.4.2　数据库操作的公共类

从软件工程的角度考虑,代码的编写要尽可能地实现重用,或者说,同样的代码要避免写两次以上。为了减少数据库操作代码的重复,将对数据库操作的共同部分提取出来,封装到一个类中(ConnDBHelper.cs)。可以通过调用类中的方法,轻松地实现数据库的操作,并把编程的精力集中在实现应用的逻辑上。该类文件存放在三层结构中的数据访问层,namespace OnLineDAL 语句把该类的命名空间定义为 OnLineDAL,与数据访问层的命名空间一致,该层中每个使用该类的文件都可引用。

类 ConnDBHelper.cs 的相关代码如下：



```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
namespace OnLineDAL
{
    public  class ConnDBHelper
    {
        private static SqlConnection connection;
        public static SqlConnection Connection
        {
            get
            {
                //直接读数据库连接信息
                //string connectionString="Data Source=.;Initial Catalog=onLineExaml;
                  User ID=sa;password=sa";
                //从 Web.config 配置文件中读入数据库连接信息
                string connectionString=ConfigurationManager.ConnectionStrings
                ["onLineExam"].ConnectionString;                //配置文件连接方式
                if (connection==null)
                {
                    connection=new SqlConnection(connectionString);
                    connection.Open();
                }
                else if (connection.State==System.Data.ConnectionState.Closed)
                {
                    connection.Open();
                }
                else if (connection.State==System.Data.ConnectionState.Broken)
                {
                    connection.Close();
                    connection.Open();
                }
                return connection;
            }
        }
        public static int ExecuteCommand(string safeSql)
        {
            SqlCommand cmd=new SqlCommand(safeSql, Connection);
            int result=cmd.ExecuteNonQuery();
            return result;
        }

        public static int ExecuteCommand(string sql, params SqlParameter[] values)
        {
            SqlCommand cmd=new SqlCommand(sql, Connection);
            cmd.Parameters.AddRange(values);
            return cmd.ExecuteNonQuery();
```



```
            }
            ...
        }
    }
```

0.5 主要模块设计

0.5.1 考生登录页面

1. 模块设计

功能：考生选择考试科目，输入学号和密码并进入考试界面开始考试，若忘记输入任何一项，会出现提示“只有学号和密码正确的考生才能进入考试”；如果已经参加该门课程的考试，则会出现相应提示“不允许进入”。具体界面参考图 0-16。

主要逻辑：Page_Load 事件中，从科目表中取得考试科目名称绑定显示在 DropDownList 控件中。

单击“开始考试”按钮时，进行如下操作。

(1) 判断考生是否合法存在。

(2) 判断该考生是否已经参加该门课程的考试。

(3) 将学号、考试科目存入 Session 中，以便传递给考试页面。

(4) 所有条件符合，转到考试页面。

2. 实现步骤

(1) 在用户界面表示层中新建一个 StudentLogin.aspx 页面。

(2) 在代码文件的头部，添加业务逻辑层和模型层的引用。

```
using OnLineBLL;
using OnLineModels;
```

(3) 为 Page_Load 事件和单击“开始考试”按钮以及查看考试成绩事件编写程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //从科目表中取得考试科目名称绑定显示在 DropDownList 控件中
        this.ddlSubject.DataSource=AllSubjectManager.GetAllSubject();
        this.ddlSubject.DataTextField="Subject_name";
        this.ddlSubject.DataValueField="Subjectt_id";
        this.ddlSubject.DataBind();
        this.ddlSubject.Items.Insert(0, new ListItem("<--请选择 -->", "0"));
    }
}
//开始考试事件
```

```
protected void btnStartExam_Click(object sender, EventArgs e)
{
    //判断学生是否存在
    if (StudentManager.Login(this.txtName.Text, this.txtPwd.Text))
    {
        string stuId=this.txtName.Text.Trim();
        string lession=this.ddlSubject.SelectedValue;
        //根据学号和考试科目判断是否已参加考试
        int subId=Convert.ToInt32(lession);
        int ret=StuScoreManager.GetStuByStuId(stuId, subId);
        if (ret>0)
        {
            Response.Write("<script>alert('你已经参加过这门课程的考试!')
            </script>");
        }
        else
        {   //将科目和学生信息保存在 Session 中,以便传递到考试页面
            Session["LoginStudent"]=stuId;
            Session["Lession"]=lession;
            Response.Redirect("~/student/startExam.aspx");
        }
    }
    else
    {
        Response.Write("<script>alert('用户名或密码错误!');</script>");
    }
}
protected void btnReset_Click(object sender, EventArgs e)
{
    this.txtName.Text=String.Empty;
    this.txtPwd.Text=String.Empty;
}
//查看已考科目成绩事件
protected void btnShow_Click(object sender, EventArgs e)
{
    String name=this.txtName.Text;
    Response.Redirect("~/student/StudentScore.aspx?sId="+name);
}
```

0.5.2 考生考试页面

1. 模块设计

功能：根据考生选择的考试科目，随机读取试卷，提交试卷时自动计算成绩，并把成绩记录到成绩表中。然后自动跳转到成绩显示页面，显示该考生的考试成绩；如果考生已经参加过其他科目的考试，在该页面显示的是参加的所有科目考试成绩。

主要逻辑：Page_Load 事件中，每种类型的试题放到一个 Repeater 控件内，从题库中根



据设置取得相应类型的试题数量,绑定到 Repeater 控件,在页面上循环显示试题。

单击“提交”按钮时,触发计算成绩事件,从试题库中取得试题和答案,对每一道试题,与取得的考生的答案对比,两者相同则在总分上加上相应题目的分数。

2. 实现步骤

(1) 在用户界面表示层中新建一个 startExam. aspx 页面。

(2) 在代码文件的头部,添加业务逻辑层和模型层的引用。

```
using OnLineBLL;
using OnLineModels;
```

(3) 为 Page_Load 事件和单击提交事件编写程序,代码略。

0.5.3 考试成绩显示页面

1. 模块设计

功能:显示考生的各门课程的成绩。

主要逻辑:在 Page_Load 事件中,首先判断考生是否登录,如果没有登录,则转向登录页面;如已登录,则根据学号显示学生的学号和姓名,并从数据库中取出该学生的所有已考科目和考试成绩,绑定到 GridView 控件中显示出来。

2. 实现步骤

(1) 在用户界面表示层中新建一个 StudentScore. aspx 页面。

(2) 在代码文件的头部,添加业务逻辑层和模型层的引用。

```
using OnLineBLL;
using OnLineModels;
```

(3) 为 Page_Load 事件编写程序,代码如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    String stuId=Request.Params["sId"];
    if (!Page.IsPostBack)
    {
        if (stuId==null)
        {
            Response.Redirect("Studentlogin.aspx");
        }
        examStudent stu=StudentManager.GetStudentById(stuId);
        this.lblStuNum.Text=stuId;
        this.lblName.Text=stu.Stu_name;
        this.gvScore.DataSource=StuScoreManager.GetAllStuScoreByStuId(stuId);
        this.gvScore.DataBind();
    }
}
```



0.5.4　考试系统后台管理登录页面

1. 模块设计

功能：系统管理员进入后台管理整个考试系统，包括管理员信息的增、删、改，班级信息的增、删、改，学生信息的增、删、改，考试科目的增、删，试题库中试题信息的增、删、改等。输入管理员账号和密码才能进入后台，若忘记输入任何一项，会进行提示。

主要逻辑：为防止在地址栏中直接输入页面地址而进入系统后台维护界面，在后台文件夹中添加一个 Web.config 文件，配置拒绝所有匿名用户访问，直接跳转到后台登录界面登录后才能访问。

2. 实现步骤

(1) 在用户界面表示层中新建一个文件夹 admin，放置后台文件，添加一个 AdminLogin.aspx 页面。

(2) 在代码文件的头部，添加业务逻辑层和模型层的引用。

```
using OnLineBLL;
using OnLineModels;
```

(3) 为进入系统后台事件编写程序，代码如下：

```
protected void btnAdminLogin_Click(object sender, EventArgs e)
{
    if (AdminManager.AdminLogin(this.txtName.Text,this.txtPwd.Text))
    {
        Session["LoginUser"]=this.txtName.Text;
        string strRedirect;
        //表单验证所指定的路径
        strRedirect=Request["ReturnUrl"];
        //表单验证
        System.Web.Security.FormsAuthentication.SetAuthCookie
        (this.txtName.Text, true);
        //登录成功后默认访问页面
        if (strRedirect==null)
            Response.Redirect("~/admin/ListAllStudents.aspx");
        //登录成功后跳转到先前访问页面
        Response.Redirect(strRedirect);
    }
    else
    {
        Response.Write("<script>alert('用户名或密码错误!');</script>");
    }
}
```



0.5.5 考试系统后台学生信息管理页面

1. 模块设计

功能：学生信息的添加、删除、修改。

主要逻辑：添加新学生时，要判断学号是否已经存在，保证学号不重复；删除学生信息时，为防止误删除，增加删除确认提示；修改学生信息时，使用隐藏控件保存并获取学生信息。

2. 实现步骤

(1) 在用户界面表示层 admin 文件夹下，添加 ListAllStudents.aspx、StudentDetails.aspx、RegisterStudent.aspx 页面。

(2) 在代码文件的头部，添加业务逻辑层和模型层的引用。

```
using OnLineBLL;
using OnLineModels;
```

(3) 判断学号是否已经存在的事件程序代码为：

```
//判断输入的学号是否已经存在!
protected void txtStuId_TextChanged(object sender, EventArgs e)
{
    string stuId=this.txtStuId.Text;
    if (StudentManager.StuIdExists(stuId))
    {
        this.lblStuId.Text="该学号已经存在,请重新编号!";
    }
    else
    {
        this.lblStuId.Text="该学号可以使用!";
    }
}
```

为删除确认事件编写程序，代码如下：

```
protected void gvStudent_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType==DataControlRowType.DataRow)
    {
        //该行的第五个单元格,增加删除确认
        e.Row.Cells[5].Attributes.Add("onclick", "return confirm('确认删除吗?')");
    }
}
```

使用隐藏控件保存并获取学生信息的处理事件，代码如下：

```
//此方法将学生所在班级读取然后显示在下拉列表框中
protected void dvStudentDetail_DataBound(object sender, EventArgs e)
{
```




```
    if (dvStudentDetail.CurrentMode==DetailsViewMode.Edit)
    {
        DropDownList ddlStudentClass=this.dvStudentDetail.FindControl
        ("ddlStudentClass")as DropDownList;
        HiddenField hfStudentClass=this.dvStudentDetail.FindControl
        ("hfStudentClass") as HiddenField;
        ddlStudentClass.SelectedValue=hfStudentClass.Value.Trim();
    }
}
protected void dvStudentDetail_ItemUpdating(object sender, DetailsViewUpdateEventArgs e)
{
    //获取学生所在班级的值
    DropDownList ddlStudentClass=this.dvStudentDetail.FindControl("ddlStudentClass")
    as DropDownList;
    //将学生班级的值添加到绑定的数据源中
    this.odsStudentDetail.UpdateParameters.Add("class_id",ddlStudentClass.
    SelectedValue);
}
```



任务1　ASP.NET概述及运行环境的构建

技能目标

会进行 ASP. NET 开发环境和运行环境的搭建，能编写和发布简单的 ASP. NET 程序。

知识目标

掌握 Visual Studio. NET 2005 集成开发环境的安装，掌握 ASP. NET 运行环境的安装与配置，理解实现动态站点的关键技术，了解 ASP. NET 程序的开发过程。

任务描述

学习一种新的动态网页技术，首先要知道它的开发环境是什么，将来开发完之后怎样运行，都需要安装什么软件。

本章任务如下：

- 安装 ASP. NET 集成开发环境——Visual Studio. NET 2005；
- 编写第一个 ASP. NET 程序；
- 搭建 ASP. NET 运行环境——IIS 服务器的安装与配置。

1.1　知 识 准 备

1.1.1　实现动态站点的关键技术

随着计算机与网络技术的发展，人们对网页的要求已经不再停留在静态网站上了，网站的动态设计成了一种必然的趋势。

目前比较流行的动态网站技术主要有 ASP、PHP、JSP 和 ASP. NET，下面分别进行简单介绍。

1. ASP

ASP 即 Active Server Pages（活动服务器页面），它是微软（Microsoft）公司开发的一种 HTML（超文本标识语言）、Script（脚本，由一组可以在 Web 服务器端或客户浏览器端运行的命令组成）与 CGI 的结合体。ASP 包含 HTML 标签，也可以直接存取数据库及使用无限扩充的 ActiveX 控件，因此在程序编制上要比 HTML 方便而且更富有灵活性。

2．PHP

PHP 即 HyperText Preprocessor(超文本预处理器),其语法借鉴了 C、Java、Perl 等语言,只需要很少的编程知识就能使用 PHP 建立一个真正交互的 Web 站点。PHP 与 HTML 语言具有非常好的兼容性,用户可以直接在脚本代码中加入 HTML 标签,或者在 HTML 标签中加入脚本代码,从而更好地实现页面控制。PHP 提供了标准的数据库接口,数据库连接方便,兼容性强,扩展性强,可以进行面向对象编程。

3．JSP

JSP 即 Java Server Pages(Java 服务器页面),它是 Sun Microsystem 公司于 1999 年 6 月推出的新技术,是基于 Java Servlet 以及整个 Java 体系的 Web 开发技术。由于 JSP 基于强大的 Java 语言,具有很强的扩展能力、良好的收缩性,以及与平台无关的开发特性,所以被许多人认为是未来最有发展前途的动态网站技术。

4．ASP.NET

在 ASP 的基础上,Microsoft 公司推出了 ASP. NET。ASP. NET 不仅借鉴了 ASP 技术的优点,而且借鉴了 Java 语言和 VB 语言的开发优势,成为 Microsoft 推出的新一代 Active Server Pages。ASP. NET 并不是 ASP 的简单升级,ASP 与 ASP. NET 主要有以下不同。

(1) 开发语言不同

ASP 使用 JavaScript 或 VBScript 脚本语言来开发,用户向 Web 页中添加 ASP 代码的方法与客户端脚本中添加代码的方法相同,导致代码杂乱。

ASP. NET 可以使用 JScript 脚本语言,也可以使用功能完善的 C♯ 及 VB. NET 编程语言,并且允许使用. NET Framework。

(2) 运行机制不同

ASP 是解释运行的编程框架,所以执行效率较低。

ASP. NET 是编译性的编程框架,运行的是服务器上的编译好的公共语言运行时库代码,可以利用早期绑定,实施编译来提高效率。

(3) 开发方式不同

ASP 把界面设计代码和业务逻辑代码混在一起,维护性和复用性困难。

ASP. NET 把界面设计和程序设计以不同的文件分离开,复用性和维护性得到了提高。

以上几种动态网页技术的比较见表 1-1。

表 1-1　动态网页技术的比较

名　称	使 用 语 言	程序脚本是否与 HTML 兼容	优　　势
PHP	C、Java、Perl 及 PHP 的新语法	是	对数据库支持好
ASP	JavaScript、VBScript	是	简单易学
JSP	Java	是	有可移植性
ASP. NET	C♯ 、VB. NET 或 JScript. NET	是	有可移植性



1.1.2 ASP.NET 介绍

作为目前比较流行的动态网站技术，ASP.NET与其他动态网页技术相比，具有以下优点。

(1) 开发工具使用方便

ASP.NET使用Visual Studio集成开发环境，Visual Studio界面友好，使用方便。

Visual Studio工具箱中包含有丰富的控件，这些控件分成不同的类别，便于查找，如图1-1所示，使用这些控件时直接拖曳控件到设计工作窗口即可，通过使用控件，可以大大减少代码的编写量。例如当需要验证用户名是否为空时，可拖曳工具箱中"验证"选项区中的RequiredFieldValidator控件到设计工作窗口，选择RequiredFieldValidator控件设置相关属性和事件即可完成该任务。



图1-1 Visual Studio工具箱

每个控件都有自己的属性、方法和事件，用于控制控件在应用程序中的外观和行为。所有ASP.NET控件和其他对象都可引发事件，可通过代码以程序方式处理这些事件，从而更好地管理代码。

(2) 将业务逻辑代码与界面设计代码分开

ASP.NET将界面设计代码和业务逻辑代码分开，界面设计代码封装在.aspx文件中，业务逻辑代码封装在.cs文件中，便于用户阅读和维护代码，并便于程序员和设计人员独立工作。在设计界面中选择某一控件，在对应事件后单击，直接能定位到.cs文件中的该事件。

(3) 易于配置和部署

ASP.NET利用基于XML的纯文本文件配置，修改配置文件后无须重新启动服务器。甚至在部署或替换运行的已编译代码时也不需要重新启动。

(4) 执行效率高

ASP.NET是将程序在服务器端首次运行时进行编译执行，以后再运行时直接执行，使得应用程序的执行效率有了很大的提高。

(5) 安全性高

ASP.NET为Web应用程序提供了默认的授权和身份验证方案。开发人员可以根据应用程序的需要很容易地添加、删除或替换这些方案。ASP.NET允许创建"用户账户"和"角色"，以便每个用户都能访问不同的代码和可执行代码，从而提高应用程序的安全性。

(6) 可以跟踪和调试

ASP.NET提供了跟踪服务，该服务可在应用程序级别和页面级别调试过程中启用。可以选择查看页面的信息，或者使用应用程序级别的跟踪查看工具查看信息。



1.2 任务实施

1.2.1 安装 Visual Studio.NET 集成开发环境

1. Visual Studio.NET 介绍

Visual Studio 是一套完整的开发工具集，用于生成 ASP.NET Web 应用程序、XML Web Services、桌面应用程序和移动应用程序。Visual Studio.NET 是一套集成开发环境(IDE)，支持 Visual Basic、Visual C++、Visual C#和 Visual J#，利用 Visual Studio 可以共享工具且有助于创建混合语言解决方案。目前使用较多的 Visual Studio.NET 版本是 Visual Studio.NET 2005。

安装 Visual Studio.NET 2005 需要满足的系统要求见表 1-2。

表 1-2　安装 Visual Studio.NET 2005 需要满足的系统要求

处理器	600MHz Pentium；推荐 1GHz Pentium 处理器
操作系统	Windows XP、Windows 2000、Windows Server 2003
内存	128MB；推荐 256MB 或更高
硬盘	不安装 MSDN：安装驱动器上要有 2GB 可用空间，系统驱动器上要有 1GB 可用空间。安装 MSDN：在完全安装 MSDN 的安装驱动器上要有 3.8GB 的可用空间；在进行默认 MSDN 安装的安装驱动器上要有 2.8GB 的可用空间，系统驱动器上要有 1GB 可用空间
显示器	800dpi×600dpi，256 色；推荐 1024dpi×768dpi，增强色为 16 位

2. 安装 Visual Studio.NET

可以从光盘安装，也可以下载安装包安装。下面以安装包为例进行说明，安装步骤如下：

(1) 双击运行安装文件 autorun.exe 或 setup.exe，运行显示界面如图 1-2 所示。



图 1-2　Visual Studio 2005 安装选择



(2) 选择“安装 Visual Studio 2005”选项，开始加载安装组件，运行显示界面如图 1-3 所示。



图 1-3　加载安装组件

(3) 单击“下一步”按钮，运行显示界面如图 1-4 所示。



图 1-4　接受用户许可协议并输入产品密钥

(4) 选中“我接受许可协议中的条款”复选框，输入产品密钥，单击“下一步”按钮，提示选择安装的功能和安装路径，运行显示界面如图 1-5 所示。

(5) 选择安装功能。这里有“默认值”、“完全”和“自定义”三种安装方式，初学者可选择“默认值”单选按钮，安装常用功能。单击“浏览”按钮设置好安装目录。单击“安装”按钮，安装程序开始安装各种组件，运行显示界面如图 1-6 所示。





图 1-5　选择安装的功能和安装路径



图 1-6　正在安装组件

（6）组件安装完成后，运行显示界面如图 1-7 所示。如果用户没有安装 Office 2003，会出现“Office 依赖项警告”提示，可不予理会。

（7）单击“完成”按钮，回到图 1-2 所示的 Visual Studio 2005 安装选择界面。如果要继续安装 MSDN，则选择“安装产品文档”选项，开始安装 MSDN；如果不安装文档，单击“退出”按钮，安装完成。

小贴士

MSDN 是微软公司对使用微软产品和技术的开发人员所设计的一项资源订阅服务，提





图 1-7　安装完成

供了丰富的技术资源，包含.NET 的文档、Visual Studio 的帮助库等内容，可以说它是学习.NET 较全的材料。很多 Visual Studio. NET 安装光盘和软件包中不包含 MSDN（约 1.5GB），如果要安装 MSDN，用户需要专门下载。

完成安装后，选择“开始”|“程序”| Microsoft Visual Studio 2005 | Microsoft Visual Studio 2005 命令，如图 1-8 所示，运行 Visual Studio 2005。



图 1-8　运行 Visual Studio 2005

首次运行，系统会提示选择默认环境设置，如图 1-9 所示。

在图 1-9 中选择“Visual C＃开发设置”选项，单击“启动 Visual Studio”按钮，系统提示为“第一次使用配置环境，可能要花几分钟”，以后再使用时不会出现此提示。然后系统显示“起始页”窗口，如图 1-10 所示。



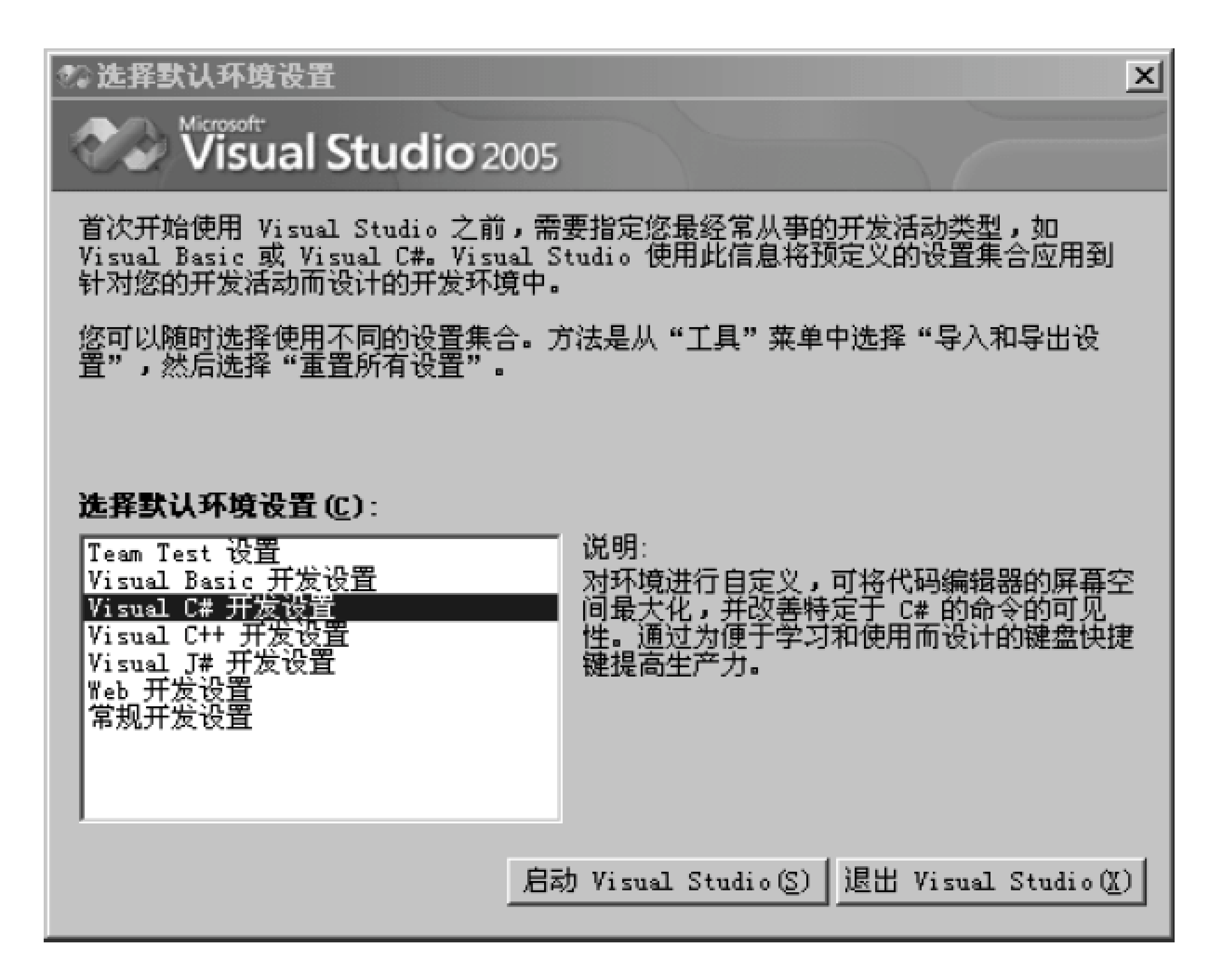


图 1-9　选择默认环境设置

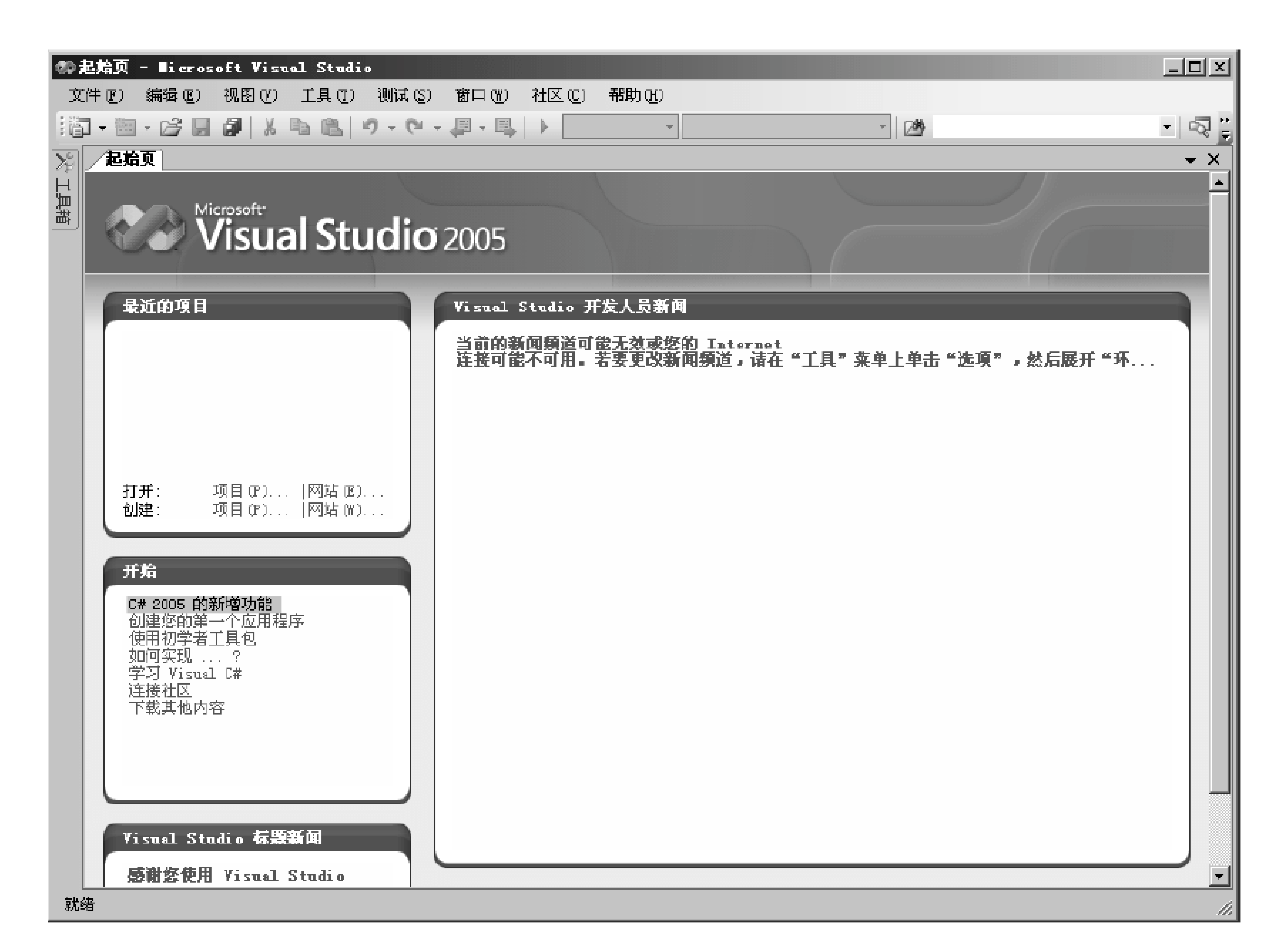


图 1-10　“起始页”窗口

1.2.2　第一个 ASP.NET 程序

1. 编写第一个 ASP.NET 程序

创建一个简单的 ASP.NET 程序，如图 1-11 所示，页面显示一幅图片和两个按钮。当单击“隐藏”按钮时，图像被隐藏；单击“显示”按钮时，图像又显现。

（1）选择“开始”|“程序”|Microsoft Visual Studio 2005|Microsoft Visual Studio 2005 命令，启动 Visual Studio 2005。

（2）选择“文件”|“新建”|“网站”命令，打开“新建网站”对话框，如图 1-12 所示。在“位置”下拉列表框中有“文件系统”、HTTP 和 FTP 三个选项，选择“文件系统”选项；在“语言”下拉列表框中选择 Visual C＃选项，单击“浏览”按钮，选择网站保存位置。

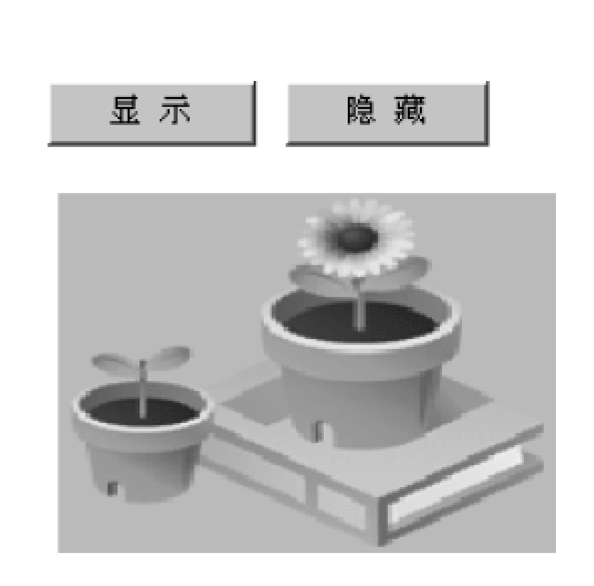




图 1-11　第一个 ASP.NET 程序



图 1-12　“新建网站”对话框

（3）单击“确定”按钮，生成一个默认名称为 Default.aspx 的网页，如图 1-13 所示，网页有“设计”和“源”两种视图，图 1-13 为源视图，显示的是页面的源代码。

（4）在图 1-13 中单击“设计”按钮，切换到设计视图。选择“视图”|“解决方案资源管理器”命令，如图 1-14 所示，打开解决方案资源管理器。再选择“视图”|“属性窗口”命令，打开属性窗口。现在还没有向页面添加内容，显示的是空白页面，如图 1-15 所示。

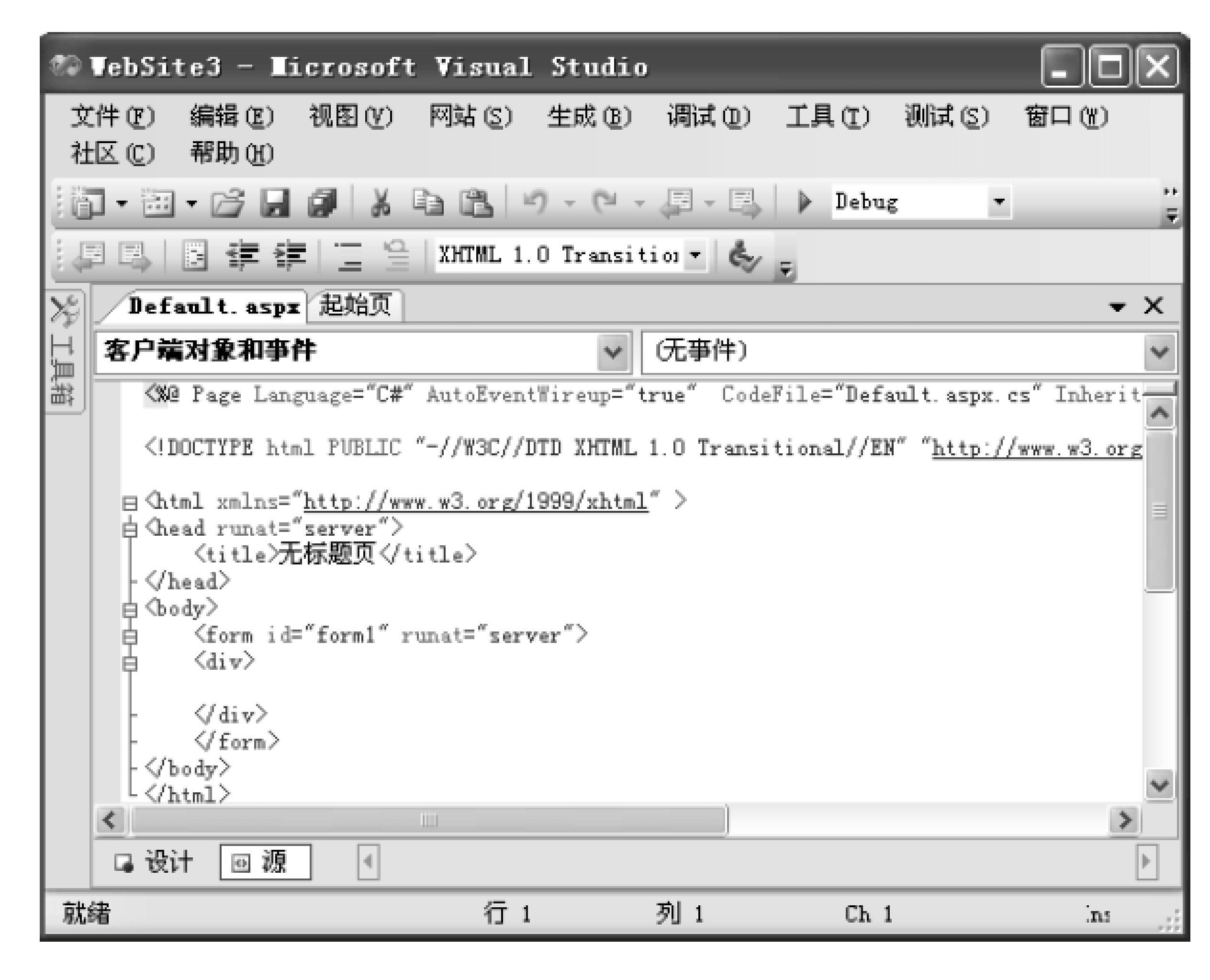


图 1-13　Default.aspx 页面的默认视图

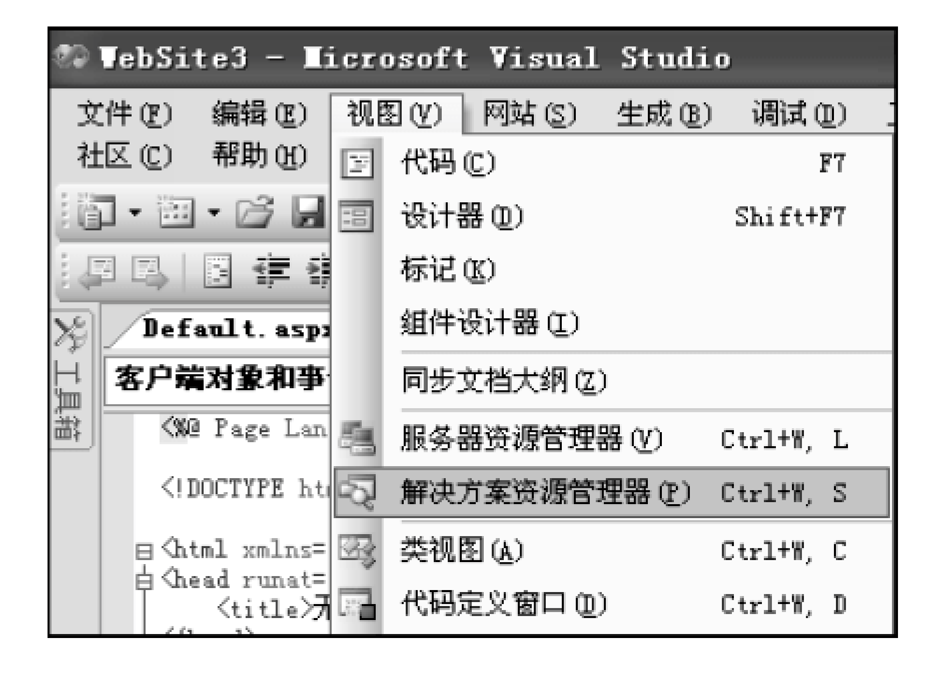


图 1-14　打开解决方案资源管理器

（5）将光标悬浮到窗口左侧的“工具箱”符号上方，显示工具箱，如图 1-16 所示。

（6）在工具箱中拖曳 Button 控件到设计视图。选中该按钮控件，在属性窗口修改它的 ID 属性值为 btnShow，Text 属性值为“显示”，用鼠标拖动修改 Button 按钮的大小。用同样的方法再添加一个 Button 按钮，设置其 ID 属性值为 btnHide，Text 属性值为“隐藏”。



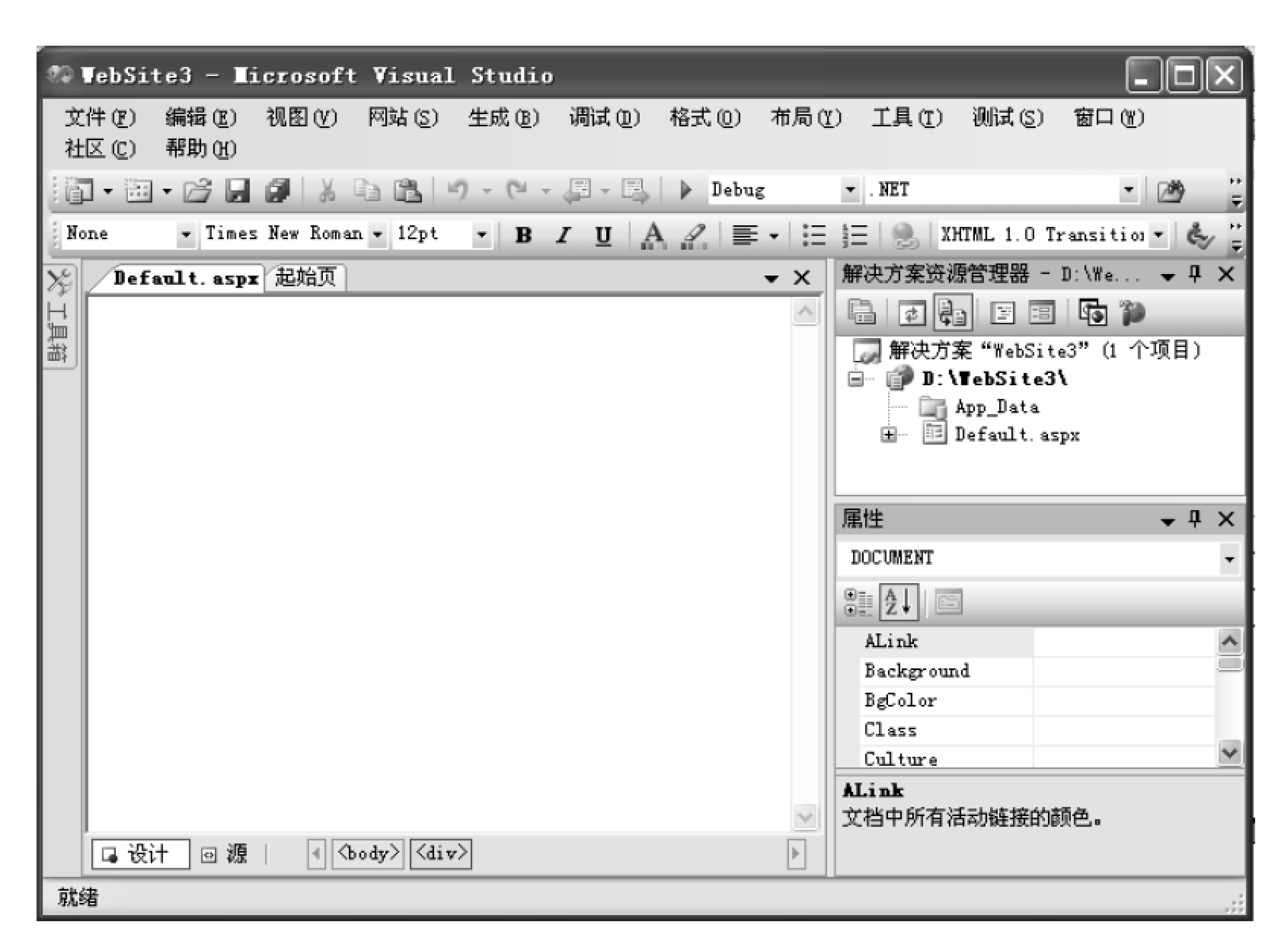


图 1-15　Default.aspx 的设计视图

(7) 拖曳 Image 控件到设计视图。选择该 Image 控件，在属性窗口修改它的 ID 属性值为 imgFlower，在 ImageUrl 属性后单击[...]按钮选择图像。设置好属性后页面显示如图 1-17 所示。

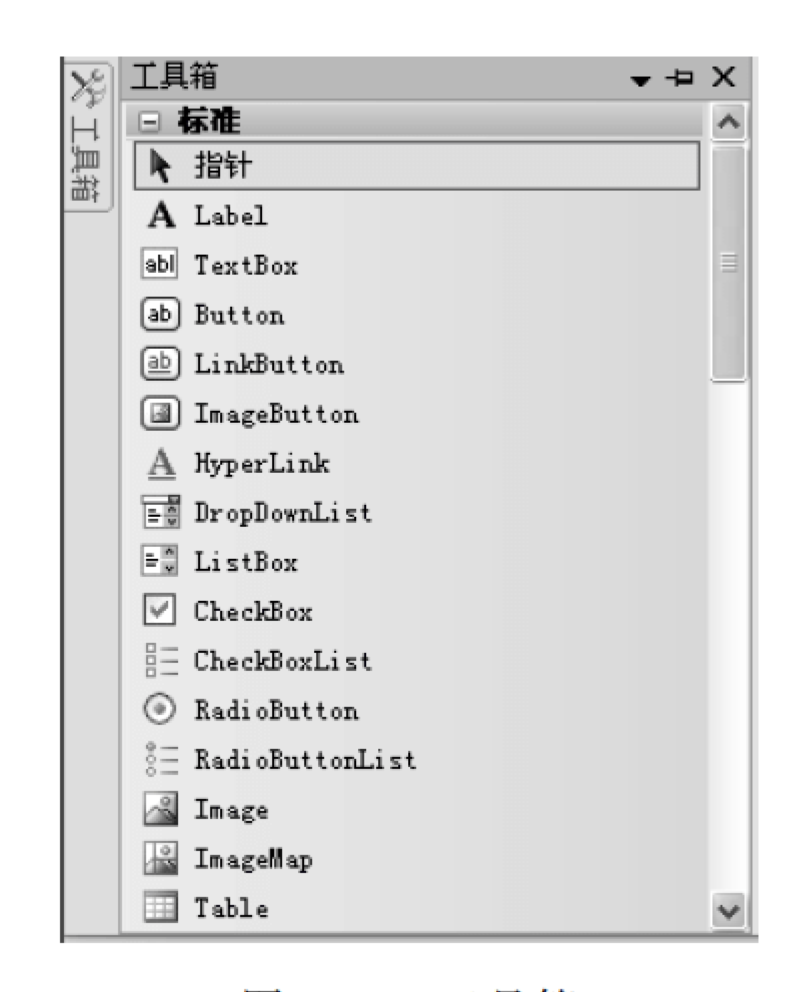


图 1-16　工具箱

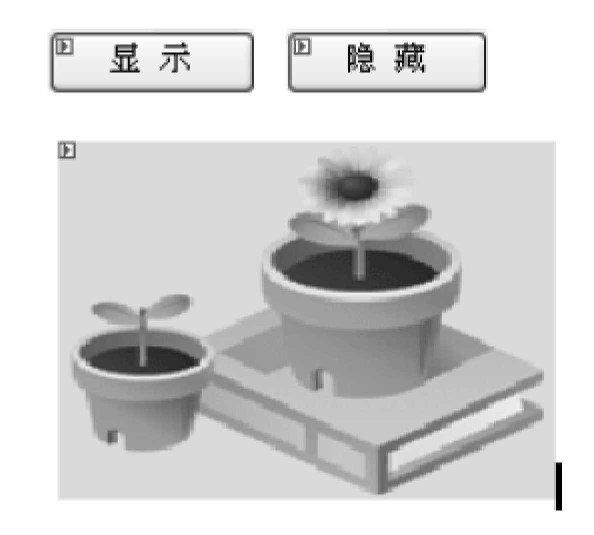


图 1-17　网页显示结果

小贴士

ID 属性是一个控件的唯一标识，就像我们人的身份证号一样。为了达到见名知意的效果，ID 命名一般由单词或单词缩写组成，第一个单词(或单词缩写)小写，后面单词(或单词缩写)首字母大写。我们上面的命名"btnHide"，btn 代表这是一个 Button 控件，Hide 代表隐藏。Text 属性值是控件上面显示的文字。

(8) 双击"显示"按钮，进入到 Default.aspx.cs 文件，并生成 btnShow_Click 事件，在里面输入代码"imgFlower.Visible=true;"。单击 Default.aspx * 标签回到 Default.aspx 文



件，双击“隐藏”按钮，进入到 Default. aspx. cs 文件，并生成 btnHide_Click 事件，在里面输入代码“imgFlower. Visible=false;”，如图 1-18 所示。

```
protected void btnShow_Click(object sender, EventArgs e)
{
    imgFlower.Visible = true;
}
protected void btnHide_Click(object sender, EventArgs e)
{
    imgFlower.Visible = false;
}
}
```

图 1-18　为按钮控件事件添加代码

(9) 选择“生成”|“生成解决方案”命令，如果没有语法错误，窗口下方会显示“生成成功”；否则，下方会提示错误，检查是否有输入错误并改正，然后重新生成解决方案。

(10) 选择“调试”|“开始执行(不调试)”命令，显示结果如图 1-11 所示。当单击“隐藏”按钮时，图像被隐藏；单击“显示”按钮时，图像又显现。

(11) 选择“文件”|“全部保存”命令，保存项目。

(12) 解决方案文件的默认存储路径和前面第(2)步指定的网站存储路径并不一致，为了方便以后打开解决方案文件，需要修改解决方案文件的位置，方法是在解决方案资源管理器中选择“解决方案”节点，此时在文件菜单中会出现“ *. sln 另存为”命令，把解决方案文件另存到网站目录中。

小贴士

Visual Studio 能自动识别各种关键字、系统函数、成员变量，自动给出输入提示，如图 1-19 所示。当输入“im”时，下面会提示出很多以 im 开始的关键字、变量名以及属性、属性值等，只需选择控件名 imgFlower；当在 imgFlower 后输入“.”后，又会给出许多提示，这里选择属性 Visible。Visual Studio 还能自动更正大小写错误、自动标示错误等，有助于提高开发过程的自动化和开发效率。

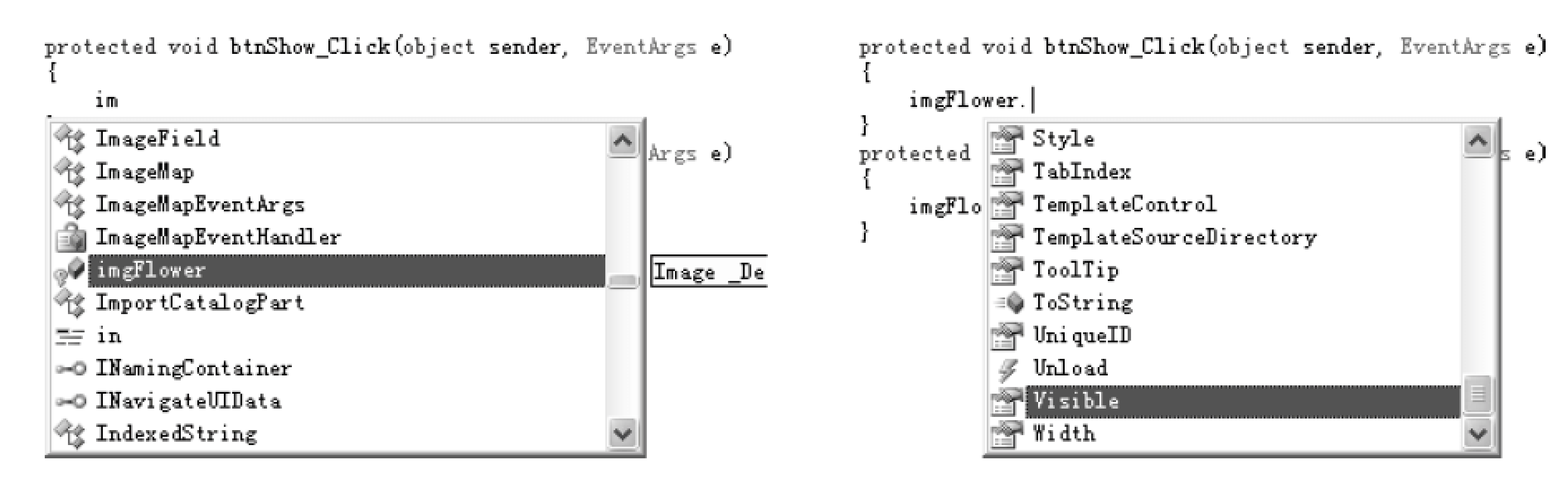


图 1-19　为按钮控件事件添加代码

2. ASP.NET 程序的发布

现在第一个 ASP. NET 程序已经能在开发环境 Visual Studio 环境下运行了，但如果要



脱离开发环境,在服务器上运行,还需要对网站进行发布。

在菜单栏选择“生成”|“发布网站”命令,如图 1-20 所示,系统显示“发布网站”对话框,如图 1-21 所示。在此对话框中选择发布后的目标位置,单击“确定”按钮。发布成功后所有的.cs 文件都被编译成 bin 文件夹下的 DLL 文件。

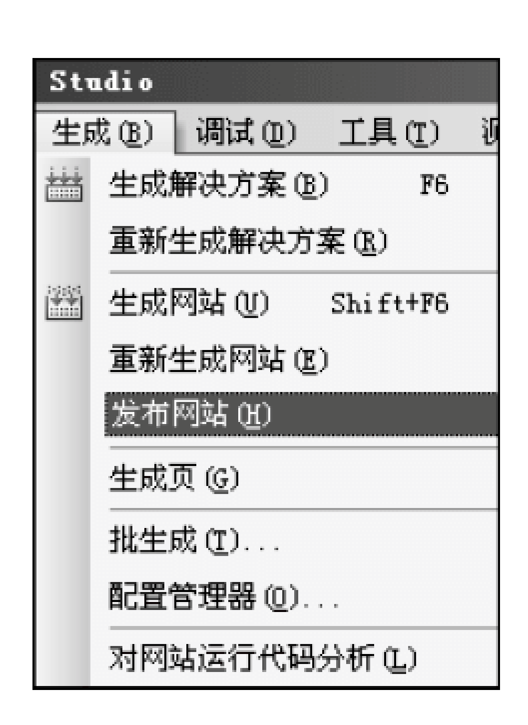


图 1-20　发布网站



图 1-21　“发布网站”对话框

1.2.3　搭建 ASP.NET 的运行环境

ASP.NET 使用 IIS 作为它的 Web 服务器,本节介绍如何进行 IIS 的安装和配置。

1. 安装 IIS 服务器

在 Windows 的安装盘中有 IIS 软件,安装 IIS 组件时把 Windows 安装盘插入光驱即可自动被寻址到,不同的 Windows 版本安装方式稍有不同。下面在 Windows Server 2003 下安装 IIS。

小贴士

如果没有操作系统安装盘,可以从网上下载 IIS 安装包。

(1) 打开“控制面板”,双击“添加或删除程序”按钮。

(2) 在弹出的窗口中单击“添加/删除 Windows 组件”按钮,显示如图 1-22 所示对话框。

(3) 在“组件”列表框中拖动垂直滚动条,选中“应用程序服务器”复选框,单击“详细信息”按钮,显示如图 1-23 所示对话框。

(4) 选中 ASP.NET 和“Internet 信息服务(IIS)”复选框,单击“确定”按钮。开始对所选组件进行安装。安装完成后显示如图 1-24 所示界面。

(5) 单击“完成”按钮退出组件向导。

2. 配置 IIS 服务器

完成 IIS 的安装后,单击“开始”|“程序”|“管理工具”|“Internet 信息服务(IIS)”命令,





图 1-22 Windows 组件向导



图 1-23 “应用程序服务器”对话框



图 1-24 安装完成



即可启动“Internet 信息服务”管理工具，如图 1-25 所示。



图 1-25　Internet 信息服务

若要启动或停止 IIS 服务，可右击“默认网站”节点，在弹出的快捷菜单中选择“启动”或“停止”命令，或者选中“默认网站”后单击工具栏上的“启动”或“停止”按钮。

安装 IIS 后，系统自动创建了一个默认的 Web 站点，该站点的主目录默认为 C:\INETpub\wwwroot。用户也可重新设定主目录，方法为：右击“默认网站”节点，在弹出的快捷菜单中选择“属性”命令，此时就可以打开“默认网站 属性”设置对话框，如图 1-26 所示。在该对话框中选择“主目录”选项卡，单击“本地路径”后面的“浏览”按钮，在“浏览文件夹”对话框中选择新的主目录，在此可以设置为前面发布本网站的目录。

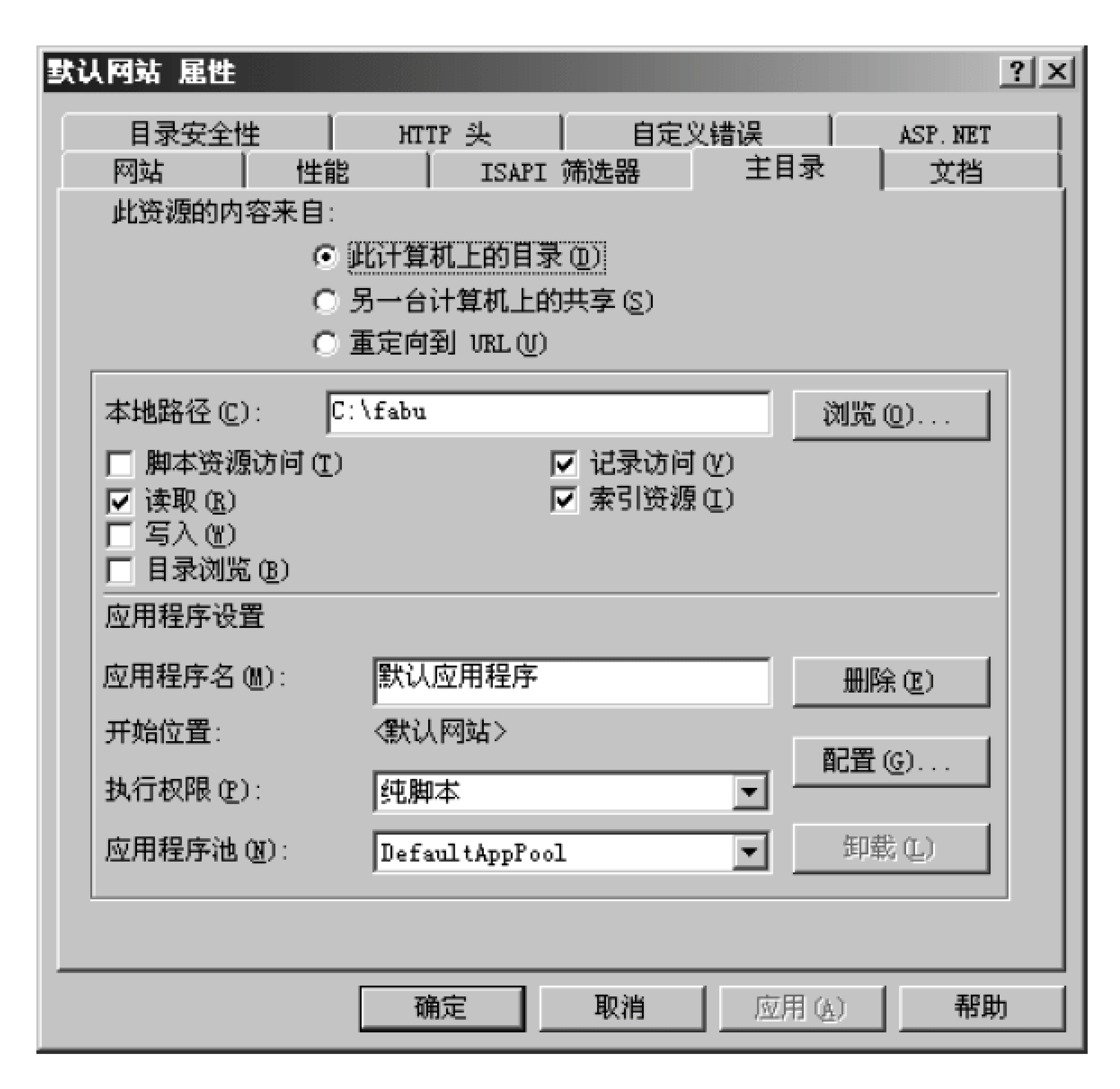


图 1-26　“默认网站 属性”设置对话框

主页文档是在浏览器中输入网站域名，而未制定所要访问的网页文件时，系统默认访问的页面文件。IIS 默认的主页文档只有 Default.htm 和 Default.asp 等，根据需要，可为站点设置所能解析的主页文档。单击“文档”选项卡，切换到对主页文档的设置页面，如图 1-27 所示，单击“添加”按钮，在弹出的“添加默认文档”对话框中输入新的默认文档名“Default.aspx”。



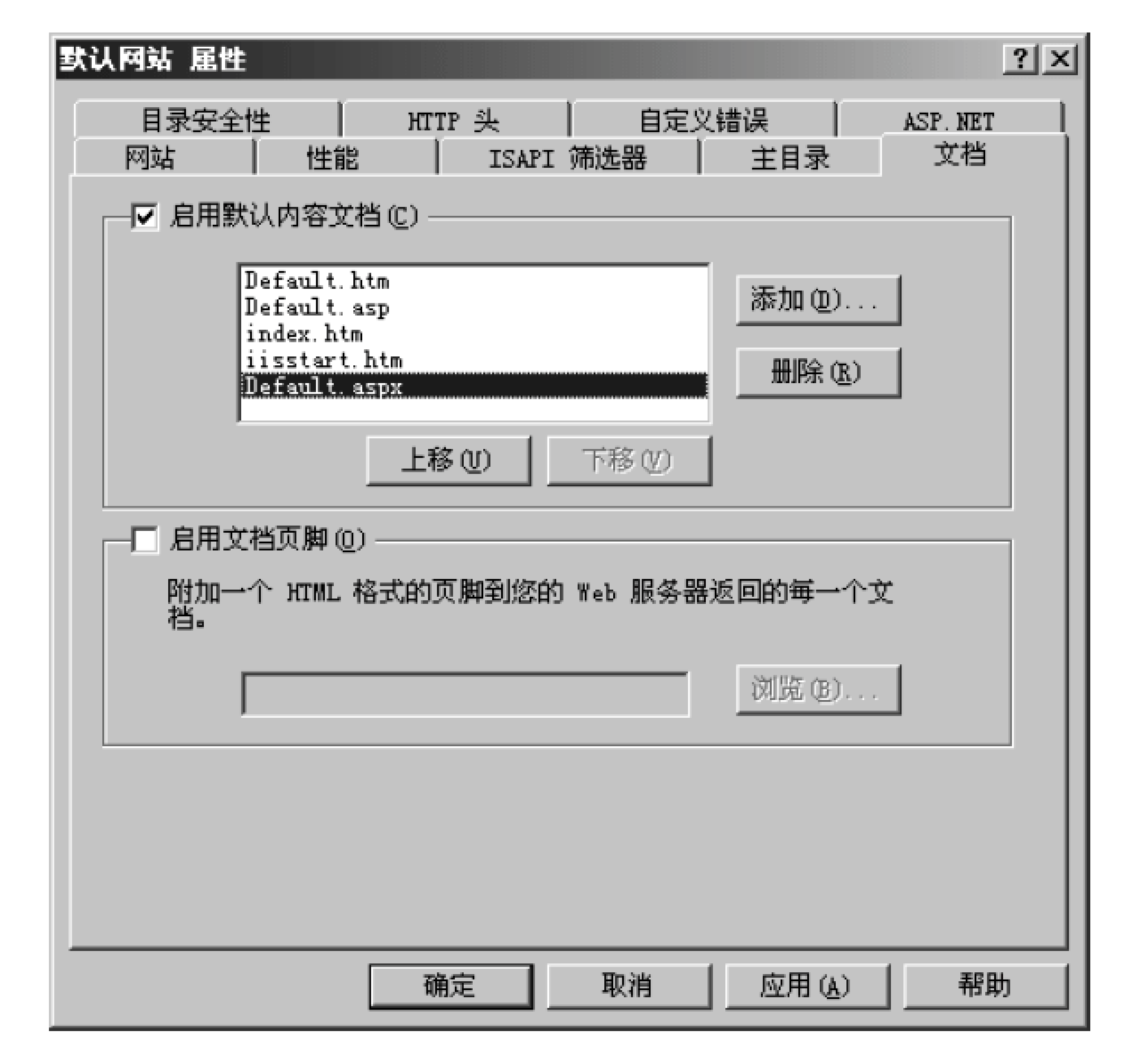


图 1-27　添加默认内容文档

安装完 IIS 后，它默认使用的 ASP.NET 版本是 1.1.4322，但 Visual Studio 2005 中使用的是 2.0.50727 版本，需要在网站属性 ASP.NET 选项卡中选择 ASP.NET 版本，如图 1-28 所示。



图 1-28　修改 ASP.NET 版本

📖小贴士

IIS 中默认 TCP 端口配置是 80 端口，如果操作系统中有其他安装程序（如迅雷）占用了 80 端口，在配置中需要修改端口，修改成未使用的端口号，如 90，如图 1-29 所示。





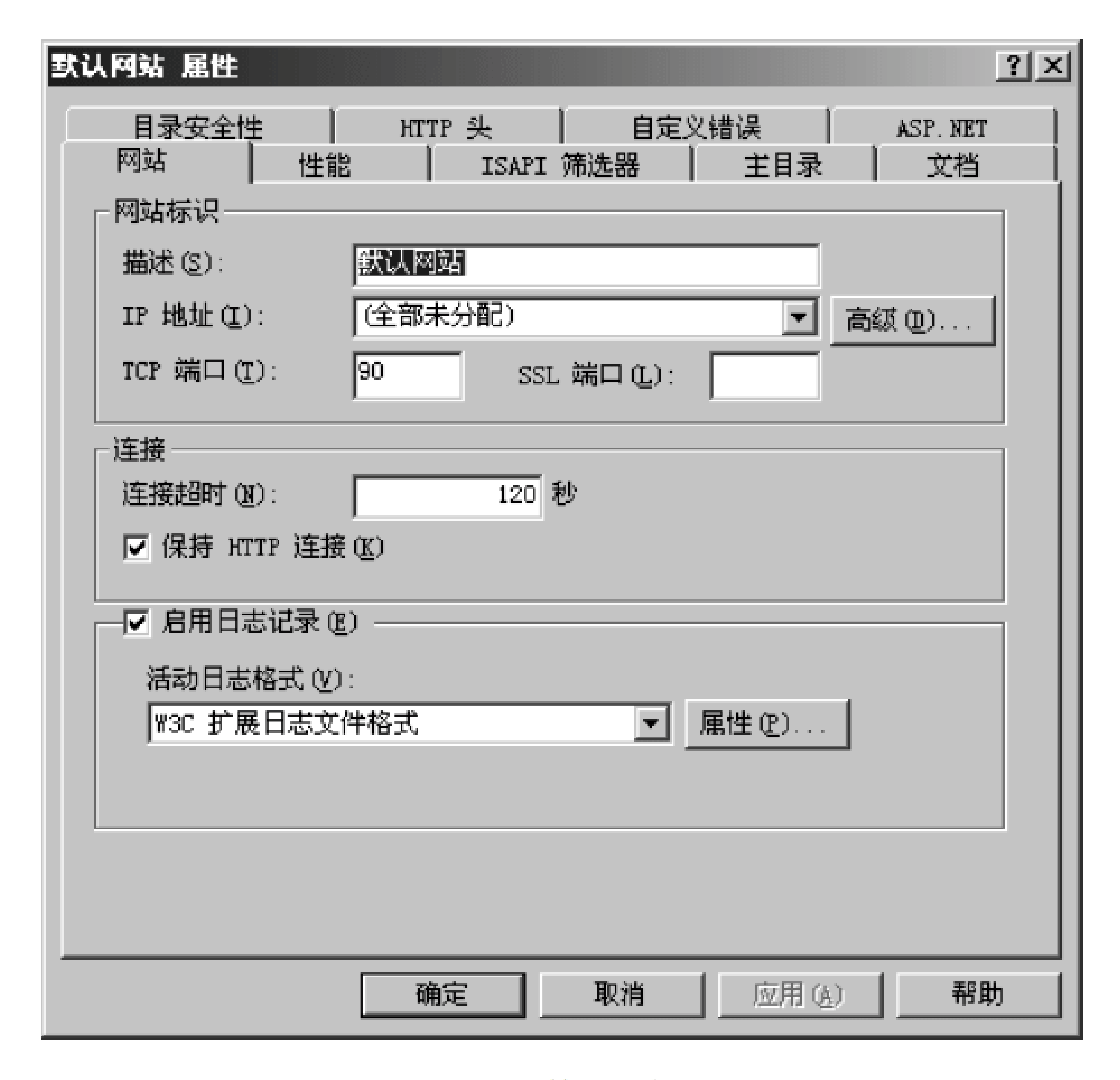


图 1-29　修改端口号

3. 测试 IIS 服务器

现在网站已经发布，IIS 已经安装配置，只需检查是否成功。在浏览器中输入“http://localhost”，可看到如图 1-30 所示内容。如果 TCP 端口号不是 80，还要加上端口号，如 http://localhost:90。



图 1-30　添加默认内容文档

小贴士

在使用 ASP.NET 制作网站的过程中，除了使用 Visual Studio 开发工具和 IIS 服务器外，做动态网站还需要安装数据库管理工具。目前常用的数据库管理工具有 Access、SQL Server、MySQL 和 Oracle，本书使用的是 SQL Server 2000。如果需要美工设计，还需要图形处理软件(如 Photoshop、Flash)和布局工具 Dreamweaver 等。本书以功能设计为主，事实上，做好网站需要学好一套课程，而不是一门课。



练　　习

1. 单项选择题

（1）ASP. NET 不能使用下面的________语言进行开发。

A. VB. NET　　B. JavaScript　　C. C#　　D. JScript. NET

（2）下面关于 ASP. NET 的介绍，________不正确。

A. 开发工具使用方便

B. 将业务逻辑代码与界面设计代码分开

C. ASP. NET 使用的开发工具是 eclipse

D. 易于配置和部署

（3）目前比较流行的动态网站技术有________。

A. ASP　　B. Dreamweaver　　C. PHP　　D. . NET

（4）ASP. NET 中，每个控件都有自己的________，用于控制控件在应用程序中的外观和行为。所有 ASP. NET 控件和其他对象都可引发事件，可通过代码以程序方式处理这些事件，从而更好地管理代码。

A. 属性　　B. 方法　　C. 事件　　D. 行为

（5）在创建 ASP. NET 网站的过程中，可以使用________工具。

A. Fireworks　　B. Visual Studio. NET

C. SQL Server　　D. Photoshop

2. 简答题

（1）ASP. NET 与 ASP 相比有什么优势？

（2）说明 ASP. NET 中的 Web 页面与其隐藏类之间的关系。

实　　训

实训目的

（1）熟练掌握 Visual Studio 2005 的启动和退出方法。

（2）熟悉 Visual Studio 2005 的集成开发环境。

实训内容

（1）启动 Visual Studio 2005。

（2）打开 Visual Studio 开发环境中解决方案资源管理器、工具箱、属性窗口和设计视图。

（3）新建一个网站 Exercise 1，做一个简单的加法器，如图 1-31 所示。其功能为：输入两个加数，单击“计算”按钮，得到和；单击“再来一次”按钮，数值清空。

（4）将上面的网站发布。







图 1-31　简单加法器

(5) 配置 IIS,在 IIS 服务器上浏览简单加法器页面。

实训指导

(1) 使用“视图”菜单,可打开解决方案资源管理器和属性窗口。将光标悬浮在工具箱图标上面可显示工具箱。单击窗口下方“设计”按钮可切换到设计视图。

(2) 简单加法器页面需要从工具箱中拖曳三个 TextBox 控件和两个 Button 控件到设计窗口上。注意控件 ID 名称,三个 TextBox 控件 ID 名称分别为 txtNum1、txtNum2 和 txtTotal,三个控件的 text 属性值都为空;两个 Button 控件的 ID 名称分别为 btnAdd 和 btnReset,两个控件的 text 属性值分别为“计算”和“再来一次”。

(3) 双击“计算”按钮,在后台程序逻辑代码文件(.cs 文件)中生成 btnAdd_Click 事件,在事件中添加代码,如图 1-32 所示。int.Parse()方法用来把字符串内容转换成整数,Convert.ToString()方法用来把数值转换成字符串。添加的语句首先把从控件中获得的字符串类型值转换成数值相加,得到的和再转换成字符串类型赋值给 txtTotal 控件的 text 属性。

```
protected void btnAdd_Click(object sender, EventArgs e)
{
    txtTotal.Text = Convert.ToString(int.Parse(txtNum1.Text) + int.Parse(txtNum2.Text));
}
```

图 1-32　求和

(4) 双击“再来一次”按钮,在后台程序逻辑代码文件(.cs 文件)中生成 btnReset_Click 事件,在事件中添加代码,如图 1-33 所示,把三个 Text 控件的值清空。

```
protected void btnReset_Click(object sender, EventArgs e)
{
    txtNum1.Text = "";
    txtNum2.Text = "";
    txtTotal.Text = "";
}
```

图 1-33　清空数值

任务2　C#语言基础与实体类创建

技能目标

会使用 Visual Studio.NET 2005 编写控制台应用程序，会创建类，能利用类的继承特性派生新的类。

知识目标

掌握 C# 语言的基本数据类型、常量和变量的概念，掌握控制语句的使用，掌握数组的相关知识，理解面向对象编程的基本思想，掌握类和对象的关系以及常用的类与函数。

任务描述

类是面向对象程序设计的灵魂，也是组成程序的最基本的模块。在本书介绍的三层架构在线考试系统中，各层之间通过实体类来传递数据，实体类的创建以及相关代码的编写是实现系统功能的基础。

本章任务如下：

- 为在线考试系统创建实体类；
- 在线考试系统倒计时的实现。

2.1　知识准备

2.1.1　C# 语言基础

1. C# 语言概述

微软公司 2000 年推出的 C#(其英文名为 C-Sharp)是从 C 语言和 C++ 语言演变而来的一种新型面向对象的编程语言，C# 语言可以编译成跨平台、跨语言的代码，它避免了 C 语言中的指针和多继承，简单易学且功能很强。2002 年微软公司推出的.NET 框架作为全新的跨语言软件开发平台，顺应了当今软件工业分布式计算、面向组件、企业级应用、软件服务化、以 Web 为中心等大趋势，成为众多软件企业的主流开发平台，并呈现出强劲的发展势头。C# 语言结合.NET 后，使 C# 语言程序设计语言成为.NET 框架的重要组成部分，也是微软公司力推的新一代程序设计语言。C# 语言＋ASP.NET 平台开发 B/S(浏览器/服务器)结构应用程序设计(与 SUN 公司的 J2EE 并行)代表了当前的主流编程方向，它越



来越多地受到人们的青睐，掌握此程序设计技术的工程师已成为企业的急需人才。

2．第一个 C#语言程序

用 C#语言在 Visual Studio 2005 中可以编写控制台应用程序、Windows 应用程序、Web 应用程序和 Web 服务等。本章将在控制台应用程序中对 C#语言的相关知识进行介绍。

创建控制台应用程序的方法如下：

启动 Visual Studio 2005，选择“文件”|“新建”|“项目”命令，打开“新建项目”对话框，如图 2-1 所示。在“项目类型”选项区中选择 Visual C#类型，在“模板”选项区中选择“控制台应用程序”选项，“名称”、“解决方案名称”使用默认值，“位置”选择“C:\Documents and Settings\Administrator\桌面”，单击“确定”按钮，即在桌面上创建了一个名称为 ConsoleApplication1 的控制台应用程序。



图 2-1　启动 Visual Studio 2005，创建控制台应用程序

在解决方案资源管理器中双击 Program.cs 类，在 Main()方法中输入代码：

```
Console.WriteLine("欢迎进入 C#世界!");
```

选择窗体菜单中的“生成”|“生成解决方案”命令，如果在窗体下方显示“生成成功”，表示输入的代码没有编译错误，如图 2-2 所示。

然后选择“调试”|“开始执行(不调试)”命令，运行程序，在控制台中显示“欢迎进入 C#世界!”，如图 2-3 所示。

小贴士

运行程序的快捷键是 Ctrl+F5。如果在代码的最后添加“Console.ReadKey();”语句，可以选择“调试”|“启动调试”命令，或按 F5 键，或单击工具栏上的“启动调试”按钮▶运行程序。运行程序后，显示的窗口可停留，按任意键结束程序。



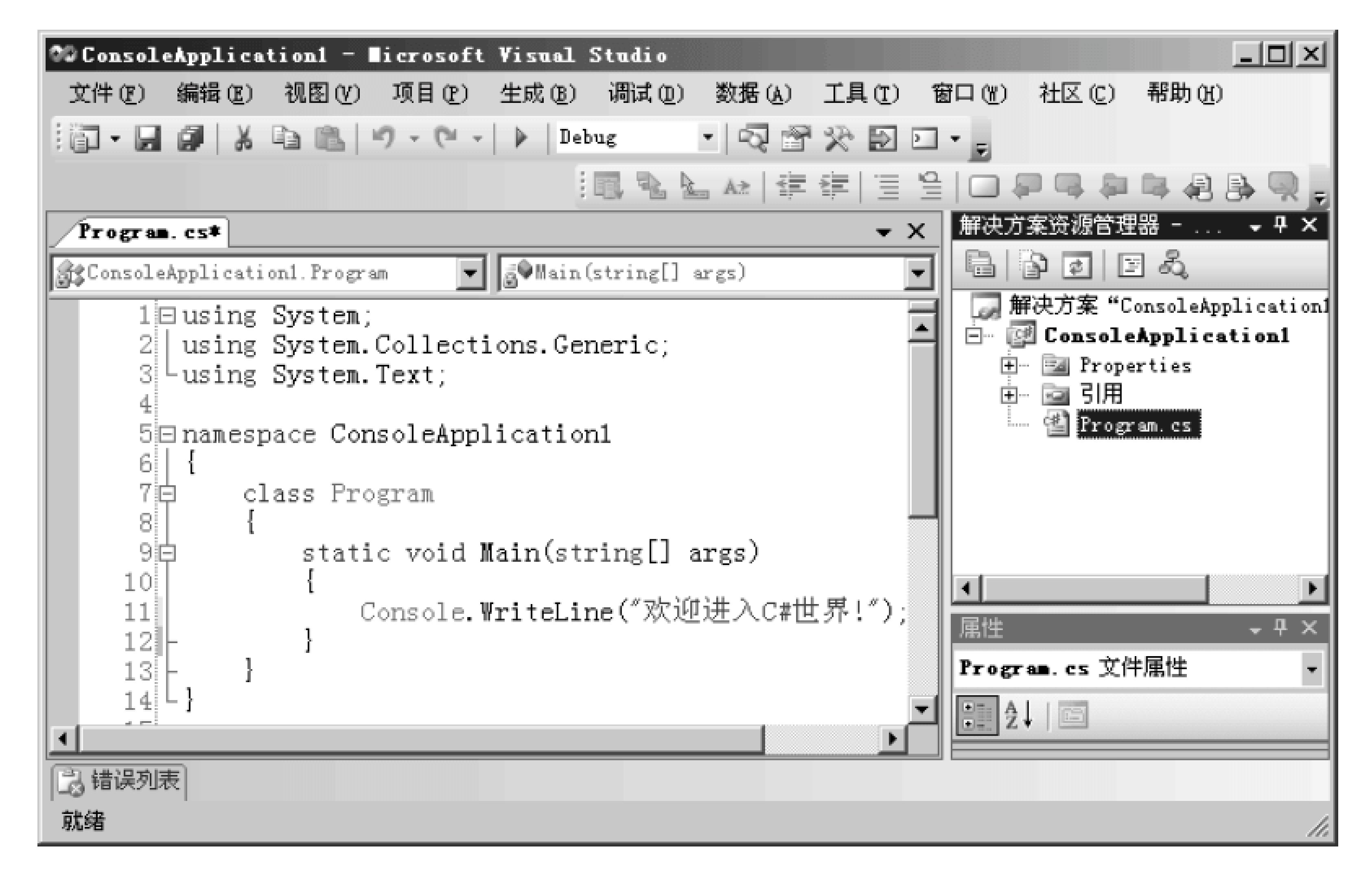


图 2-2　生成解决方案



图 2-3　运行程序

3. 认识 C#语言程序

创建控制台应用程序后，在解决方案资源管理器中，可以看到创建的解决方案、项目和类。默认情况下解决方案的名称和创建的第一个项目的名称相同，可以重命名。一个解决方案中可以包含多个项目，一个项目中又可以包含多个类。Program.cs 类是创建项目后自动添加的第一个类，双击 Program.cs 类文件，在代码编辑区显示该文件的内容。其中创建项目时自动添加的代码包括以下几种。

(1) namespace 关键字

namespace(命名空间)是 C#语言中组织代码的方式。.NET Framework 提供了一个功能强大的类库，使用命名空间可以把功能相关的一些类放在一起。命名空间包括.NET 框架定义的命名空间和用户自定义的命名空间，namespace 关键字用于用户自定义命名空间。默认情况下，命名空间的名称和类所在项目的名称一致，也可以重命名。

(2) using 关键字

using 关键字用来根据需要引用不同的命名空间，从而调用该命名空间中的类以及类中的方法。例如 System、System.Collections.Generic、System.Text 等为.NET 框架定义的命名空间，System 命名空间中包含 Console 类，Console 类中又包含 WriteLine()方法。使用 using System 语句引用命名空间 System，才能使用 Console.WriteLine()语句调用



Console 类中的 WriteLine()方法。

在编写程序时，也可以不引用命名空间，而在类的前面加上命名空间的名称，如 System.Console.WriteLine()。但在命名空间较长特别是重复使用时会加大代码的书写量。由此可见，在程序开头使用 using 关键字引用命名空间可以简化语句行的书写。

(3) class 关键字

class 关键字用来定义类。class Program 表示 Program 是程序的一个类。每个程序至少包含一个类。Program 是创建项目时自动添加的一个类，可以给这个类重命名，也可以添加新的类。

(4) Main()方法

一个项目中可以包含多个类，一个类中又可以包含多个方法。但一个项目中只能有一个 Main()方法，它是程序的入口点。

小贴士

如果看不到解决方案资源管理器，可以选择“视图”|“解决方案资源管理器”命令，快捷键为 Ctrl+Alt+L；如果在解决方案资源管理器中看不到解决方案，可以手动设置，方法为选择“工具”|“选项”|“项目和解决方案”命令，在出现的对话框中选中“总是显示解决方案”复选框。

2.1.2 C#语言中的变量和运算符

1. C#语言中的数据类型

数据类型是数据的表现和存储形式。数据类型有两个作用：①在生成对象时，它指出应为对象分配多大的存储空间；②它规定了对象所能进行的操作。C#语言中的数据类型分为两个基本类型，即值类型和引用类型。其中值类型又分为基本数据类型、枚举类型和结构类型；引用类型分为类、接口和数组。C#语言的数据类型如表 2-1 所示。

表 2-1 C#语言的数据类型

C#语言数据类型	值类型	基本数据类型
		枚举类型
		结构类型
	引用类型	类
		接口
		数组

值类型和引用类型的基本区别在于它们在内存中的存储方式，值类型直接在栈中存储该类型的值，而引用类型则在托管堆中存放实际对象，在栈中存放指向实际对象的指针。值类型的优点是访问性能好，而引用类型的优点是回收站会自动管理分配在托管堆中的内存。一般值类型用于存储数据，而引用类型用于定义行为。本章将对值类型中的基本数据类型和引用类型中的类和数组进行介绍，表 2-2 列出了 C#语言中常用的基本数据类型。

2. C#语言中的常量和变量

1) 常量

在程序中一经定义就不能改变的量叫做常量。常量是指程序运行时其值不能改变的量。



表 2-2 C#语言中常用的基本数据类型

类　型	说　明	数 据 范 围	数据示例
int	整型	－2147483648～2147483647	21
float	单精度	$\pm1.5\times10^{-45}\sim\pm3.4\times10^{38}$	21.23F
double	双精度	$\pm5.0\times10^{-324}\sim\pm1.7\times10^{308}$	21.23D
char	字符型	Unicode 字符	A
string	字符串	Unicode 字符串	ABC
bool	布尔型	true 和 false	true

常量的定义：

```
const 类型 变量名=值;
```

例如：

```
const double PI=3.14;                    //定义 PI 为字符串类型的常量
```

2）变量

变量是指程序在运行时其值可以改变的量。使用变量的原则是先定义再使用。

（1）变量的命名规则

① 可包含字母、数字、下划线；

② 不能以数字开头；

③ 不能是 C#语言的关键字；

④ 尽量能见名知意。

（2）变量的定义

```
数据类型 变量名称;
```

例如：

```
string myName;
int age;
```

（3）变量的赋值

```
变量名称=值;
```

例如：

```
myName="Lili";
age=19;
```

（4）变量的声明与赋值

```
数据类型 变量名称=值;
```

例如：

```
string myName="Lili";
```



```
int age=19;
```

(5) 变量类型的转换

对于数值类型,取值范围小的类型可以隐式转换为取值范围大的类型,如 int 类型可以隐式转换为 float、double 类型,float 可以隐式转换为 double 类型;反之则需要显式转换。变量类型的转换示意图如图 2-4 所示。

隐式转换 →
int、float、double
← 显示转换

图 2-4　变量类型的转换

例如:

```
int a=12;
double d=21.3;
double e;
e=d+a;                          //a 可以隐式转换为 double 类型
```

例如:

```
int a=12;
double d=21.3;
int e;
e=(int)d+a;                     //d 需要显式转换为 int 类型
```

数字字符串转换为数值可采用 Parse()方法。

例如:

```
string s="123456";
int i=int.Parse(s);
double j=double.Parse(Console.ReadLine());
```

数值转换为字符串可采用 ToString()方法。

例如:

```
int i=18;
double d=23.45;
string s1=i.ToString();
string s2=d.ToString();
```

使用 Convert 类的类型转换方法可以在各种基本类型之间转换。

常用的方法有:

```
Convert.Toint32()               //转换为 int 型
Convert.ToSingle()              //转换为 float 型
Convert.ToDouble()              //转换为 double 型
Convert.ToString()              //转换为 string 型
```

例如:

```
string s="123456";
int i=Convert.ToInt32(s);       //把 s 转换为 int 型
```

3) 注释

注释常被用来提供关于程序清单的描述性信息,以使程序具有可读性。关键的语句要使用注释,如变量的声明、条件判断、循环等。注释语句不参与程序的执行。



C#语言有以下三种类型的注释语句。

(1) 单行注释

格式为:

```
//注释内容
```

用两个斜杠表示注释的开始,直到该行的结尾注释结束。

(2) 多行注释

格式为:

```
/* 注释内容第 1 行
...
注释内容第 n 行 */
```

从"/ * "开始到" * /"结束,其中的内容为注释的内容。

(3) 文档注释

格式为:

```
///<summary>
///说明内容
///</summary>
```

一般在类名前应使用文档注释,说明类的功能和使用方法。

小贴士

当光标移动到某语句行,按 Ctrl+K(或 C)组合键可注释该语句行;按 Ctrl+K(或 U)组合键可取消注释;按 Ctrl+Shift+L 组合键可删除该语句行。

3. C#中的运算符和表达式

运算符是表示进行某种运算的符号;表达式是计算求值的基本单位,它是由运算符号和操作数组成的式子。操作数可以是任何类型的常量、变量、函数返回值,或者由另一个操作数和运算符组成的式子。执行表达式所规定的运算,所得到的结果值便是表达式的值。

(1) 运算符

C#语言中的运算符包括算术运算符、比较运算符、条件运算符、赋值运算符和逻辑运算符,如表 2-3 所示。

表 2-3 C#语言中的运算符

类　别	运　算　符	类　别	运　算　符
算术运算符	+ - * / % ++ --	赋值运算符	= += -= *= /= %=
比较运算符	> < >= <= !=	逻辑运算符	&& ‖ !
条件运算符	?:		

其中条件运算符"?:"为三目运算符,其基本格式为:

```
条件?成立时的值:不成立时的值
```



例如：

```
int a=3;
int b=4;
string c;
c=a>b ?"a>b" : "a<b";
Console.WriteLine(c);                          //显示 a<b
```

(2) 表达式

表达式是常量、变量、函数等用运算符连接的式子。例如 a+b、Math. Sqrt(16)、a+b=c、s="abc"+Console. ReadLine()等都可以称为表达式。

在使用 C#语言运算符连接表达式时，需要注意参与运算的操作数的类型与运算符相匹配。

2.1.3　C#语言中的 Console 类

. NET Framework 提供了内容丰富、功能强大的类库，System 命名空间中 Console 类的 WriteLine()和 ReadLine()等方法对从控制台读取字符和向控制台写入字符提供基本支持，是控制台应用程序输出语句和输入语句的关键。

1. 控制台输出语句

常用的输出方法有两个：Console. WriteLine()和 Console. Write()，它们唯一的区别是前者在输出后换行，后者在输出后不换行。

Console. WriteLine()方法输出有 3 种方式。

① Console. WriteLine()　//换行，相当于插入一空行

② Console. WriteLine(要输出的内容)　//输出内容后换行

例如：

```
Console.WriteLine("欢迎进入 C#世界!");
```

③ Console. WriteLine("格式字符串",变量列表)　//输出格式字符串后换行

例如：

```
string myName="Lili";
int age=19;
Console.WriteLine("姓名{0},年龄{1}.",myName,age);
```

显示结果：姓名 Lili，年龄 19。

其中，{0}、{1}为占位符。占位符从 0 开始向后排列，占位符的个数要和后面变量的个数相等，并且位置对应。

2. 控制台输入语句

控制台输入语句用于把用户通过键盘输入的信息读入内存。常用的输入语句有 Console. Read()、Console. ReadKey()和 Console. ReadLine()。

(1) Console. Read()语句

返回用户输入的第 1 个字符的 ASCII 码，为 int 型。



例如：

```
int i=Console.Read();
Console.WriteLine(i.ToString());                    //或 Console.WriteLine(i)
```

运行后输入“A”，按 Enter 键显示 65。

(2) Console. ReadKey()语句

等待键盘输入才退出程序，为了使调试时看到输出结果。

(3) Console. ReadLine()语句

返回用户输入的字符串，为 string 型。

例如：

```
Console.WriteLine("请输入你的名字：");
string username=Console.ReadLine();
Console.WriteLine("欢迎{0}",username);
```

运行后输入“Lili”，按 Enter 键显示“欢迎 Lili!”。

例 2-1　编写控制台应用程序，提示用户输入圆半径后，计算并显示圆的面积和周长。

```
const double PI=3.14;
Console.WriteLine("请输入圆的半径：");
double r=double.Parse(Console.ReadLine());          //需要转换为 double 类型
double L=2 * PI * r;
double S=PI * r * r;
Console.WriteLine("圆的周长为{0},面积为{1}", L, S);
```

2.1.4　C# 语言中的控制语句

1. 选择语句

当程序中需要进行两个或两个以上的选择时，可以根据条件判断来选择将要执行的一组语句。C# 语言提供的选择语句有 if 语句和 switch 语句。

1) if 语句

```
if(表达式 1)
{
    语句块 1
}
else if(表达式 2)
{
    语句块 2
}
else if(表达式 3)
{
    语句块 3
}
else
{
    语句块 n
```

```
}
```

使用说明：

(1) if 语句可同时省略 else if 和 else 语句块。

例 2-2　当用户输入的成绩大于 60 时显示“及格”。

```
Console.Write("请输入你的成绩：");
double score=double.Parse(Console.ReadLine());
if (score>=60)
{
    Console.WriteLine("及格");
}
```

(2) 可只省略 else if 语句块。

例 2-3　根据用户输入的成绩显示“及格”或“不及格”。

```
Console.Write("请输入你的成绩：");
double score=double.Parse(Console.ReadLine());
if (score>=60)
{
    Console.WriteLine("及格");
}
else
{
    Console.WriteLine("不及格");
}
```

(3) else if 语句块可以有多个。

例 2-4　根据用户输入的成绩显示不同的等级。

```
Console.Write("请输入你的成绩：");
double score=double.Parse(Console.ReadLine());
if ( score>=90)
{
    Console.WriteLine("优秀");
}
else if (score>=80)
{
    Console.WriteLine("良好");
}
else if (score>=60)
{
    Console.WriteLine("及格");
}
else
{
    Console.WriteLine("不及格");
}
```

(4) if 语句虽可包含多个语句块，但只能执行一个。当两个或多个语句块同时满足条件时，执行最前面满足条件的语句块。



(5) if 语句可以嵌套。

例 2-5 根据用户输入的成绩显示不同的等级，当输入的成绩超出范围时提示错误。

```
Console.Write("请输入你的成绩：");
double score=double.Parse(Console.ReadLine());
if (score>=0 && score<=100)
{
    if ( score>=90)
    {
        Console.WriteLine("优秀");
    }
    else if (score>=80)
    {
        Console.WriteLine("良好");
    }
    else if (score>=60)
    {
        Console.WriteLine("及格");
    }
    else if (score< 60)
    {
        Console.WriteLine("不及格");
    }
}
else
{
    Console.WriteLine("成绩输入错误!");
}
```

2) switch 语句

```
switch(控制表达式)
{
    case 常量表达式 1:
      语句 1;
      break;
    case 常量表达式 2:
      语句 2;
      break;
    case 常量表达式 n:
      语句 n;
      break;
      …
    default:
      语句;
      break;
}
```

程序执行时，先计算 switch 语句括号内控制表达式的值，再用此值和各 case 语句后的常量表达式进行比较，当某个常量表达式的值和控制表达式的值相等时，执行该常量表达式

后面的语句。如果所有常量表达式的值和控制表达式的值都不相等,则执行 default 语句后面的语句。

switch 语句后面括号中的表达式和 case 语句后面的值可以是 int、string、char 或者枚举类型。case 语句块可以根据需要添加多个。C#语言要求每个 case 语句块和 default 语句块中都必须有break 语句,除非两个 case 语句中间没有其他语句,前一个 case 语句可以不包含 break 语句。case 语句也可以和 if...else...等其他语句相互嵌套使用。

例 2-6 使用 if...else...语句嵌套 case 语句,根据用户输入的成绩显示不同的等级,当输入的成绩超出范围时提示错误。

```
Console.Write("请输入你的成绩:");
int score=int.Parse(Console.ReadLine());
if (score>=0 && score<=100)
{
    switch (score/10)
    {
        case 10:
        case 9:
            Console.WriteLine("优秀");
            break;
        case 8:
            Console.WriteLine("良好");
            break;
        case 7:
        case 6:
            Console.WriteLine("及格");
            break;
        default:
            Console.WriteLine("不及格");
            break;
    }
}
else
{
    Console.WriteLine("成绩输入错误!");
}
```

2. 循环语句

循环语句可以实现程序的重复执行。C#语言提供了 4 种循环语句,分别为 while 循环语句、do...while 循环语句、for 循环语句和 foreach 循环语句。

(1) while 循环语句

while 循环语句的语法格式为:

```
while(条件表达式)
{
    语句
}
```



程序执行时,先判断条件表达式:当值为真时,执行语句,直到条件表达式的值为假时,退出循环。

例 2-7 计算 1+2+3+…+100 值。

```
int sum=0;
int i=1;
while (i<=100)
{
    sum+=i;
    i++;
}
Console.WriteLine("1+2+3+…+100={0}", sum);
```

(2) do…while 循环语句

do…while 循环语句与 while 循环语句的功能相近,不同的是,do…while 循环语句至少执行一次内嵌的语句。其语法格式为:

```
do
{
    语句
}
while(条件表达式)
```

程序运行后,先执行一次语句,再判断条件表达式:当值为真时,继续执行语句,直到条件表达式的值为假时,退出循环。

例 2-8 使用 do…while 循环语句计算例 2-7 的值。

```
int sum=0;
int i=1;
do
{
    sum+=i;
    i++;
}
while (i<=100);
Console.WriteLine("1+2+3+…+100={0}", sum);
```

(3) for 循环语句

for 循环语句的语法格式为:

```
for(初始语句,条件语句,增/减语句)
{
    语句
}
```

for 循环语句首先执行初始语句,初始语句仅执行一次。然后判断条件语句,如为真,则执行花括号中的语句,接着执行增减语句。下次循环从条件语句开始,直到条件语句为假,退出循环。

例 2-9 使用 for 循环语句计算例 2-7 的值。



```
int sum=0;
for (int i=1; i<=100; i++)
{
    sum+=i;
}
Console.WriteLine("1+2+3+…+100={0}", sum);
```

(4) foreach 循环语句

foreach 循环语句的语法格式为：

```
foreach(类型 元素(变量名) in 集合或数组)
{
    语句
}
```

foreach 循环语句是 C# 语言中独有的，它对于处理数组和集合等数据类型的运算特别简便。一般用于遍历整个集合或数组中的元素，并且通过执行循环体对每一个元素进行操作。例如字符串可以看做是字符的集合，遍历字符串中字符，可以使用 foreach 循环语句。

例 2-10　遍历字符串中的每一个字符，显示在屏幕上。

```
Console.Write("请输入一个字符串:");
string Line=Console.ReadLine();
foreach (char c in Line)
{
    Console.WriteLine(c);
}
```

如输入“abc”后按 Enter 键，则显示结果为：

```
a
b
c
```

(5) 循环嵌套

用 Console.Write("*")语句可以在屏幕上显示一个“*”；把该语句放在循环内，多次循环后，可以显示一行“*”；把完整的循环语句块再放入一个循环内，可以显示多行“*”。这种循环方式叫做循环嵌套。

例 2-11　用循环嵌套显示 4 行、每行 5 个“*”，结果如图 2-5 所示。

```
*****
*****
*****
*****
```

图 2-5　循环嵌套

```
for (int i=1; i<5; i++)
{
  for (int j=1; j<6; j++)
  {
      Console.Write("*");
  }
  Console.WriteLine();
}
```

注意：外层循环决定“*”的行数；内层循环决定每行中“*”的个数。在内层循环后要加入 Console.WriteLine()语句用于换行。



例 2-12 使用 for 循环嵌套在屏幕上显示乘法口诀表。

```
Console.WriteLine("乘法口诀表");
for (int i=1; i<10; i++)
{
  for (int j=1; j<=i; j++)
  {
      Console.Write("{0} * {1}= {2,-5}", j, i, i * j);
  }
  Console.WriteLine();
}
```

Console.Write("{0} * {1}={2,−5}", j, i, i * j)语句中的三个占位符分别和后面的 i、j 和 i * j 对应，其中第三个占位符中的“−5”表示 i * j 的显示左对齐，占 5 个字符的位置。

在例 2-12 中，循环嵌套后，一共有两层循环，也称二重循环。根据解决实际问题的需要，也可以使用三重循环、四重循环等。

例 2-13 用 50 元、20 元、10 元和 5 元中的 1～4 种面值组成 100 元有几种情况？

可以采用四重循环来求解：

```
Console.WriteLine("50元\t20元\t10元\t5元");
Console.WriteLine("---------------------------");
for (int i=0; i<=2; i++)                              //50元
{
    for (int j=0; j<=5; j++)                          //20元
    {
        for (int k=0; k<=10; k++)                     //10元
        {
            for (int m=0; m<=20; m++)                 //5元
            {
                if (i * 50+j * 20+k * 10+m * 5==100)
                {
                    Console.WriteLine(" {0}\t{1}\t{2}\t{3}", i, j, k, m);
                }
            }
        }
    }
}
```

运行结果如图 2-6 所示。

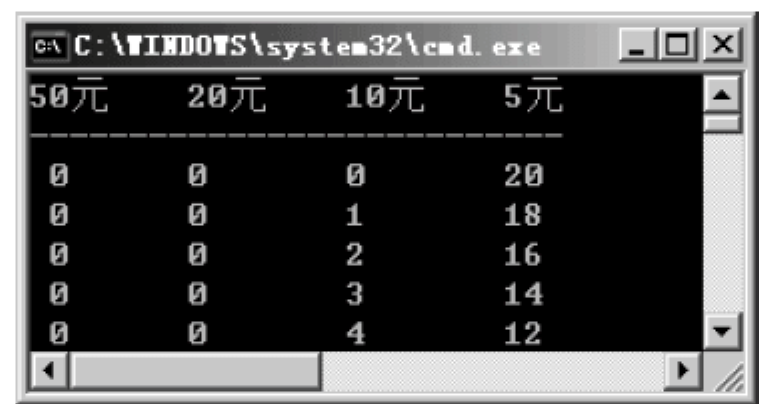


图 2-6 组成 100 元的各种情况

2.1.5 数组

数组就是一系列有序数据的集合，它的每一个元素都必须属于同一种数据类型。对每个元素，由一个统一的数组名和不同的下标来确定，数组元素的下标从 0 开始。

1. 一维数组

一维数组是指用一个下标来区分数组中各元素的数组。例如用序号来确定一个同学，全班同学就组成一个一维数组。



1）一维数组的定义

数据类型[] 数组名=new 数据类型[长度]

例如：

```
int[] myArray=new int[5];
```

2）一维数组的赋值

假设给数组 myArray 赋值为 1、2、3、4、5。可以在定义时赋值；也可以使用赋值语句赋值；因为各元素的值等于元素的下标加 1，有一定的规律，所以也可以采用循环赋值。

（1）定义时赋值

```
int[] myArray=new int[5] {1, 2, 3, 4, 5};
```

（2）使用赋值语句

```
int[] myArray=new int[5];
myArray[0]=1;
myArray[2]=3;
…
```

（3）使用循环赋值

```
int[] myArray=new int[5];
for (int i=0; i<myArray.Length; i++)
{
    myArray[i]=i+1;
}
```

3）一维数组的显示

```
for (int i=0; i<myArray.Length; i++)
{
    Console.Write("{0}  ",myArray[i]);
}
Console.WriteLine();
```

也可采用 foreach 循环进行显示：

```
foreach (int i in myArray)
{
    Console.Write(i+"  ");
}
Console.WriteLine();
```

2. 多维数组

多维数组是指用多个下标来区分数组中各元素的数组。例如用座位的行号、列号来确定一个同学，全班同学就组成一个二维数组。每个同学是二维数组中的一个元素；而用班号、行号、列号来确定一个同学，多个班的同学又组成一个三维数组，每个同学是三维数组中的一个元素；等等。



1）多维数组的定义

多维数组在定义时下标的个数要和维度一致。

例如：

```
int[ , ] myArray=new int[3,4];                    //定义一个二维数组
int[ , ,] myArray=new int[10, 6,5];              //定义一个三维数组
```

2）多维数组的赋值

以二维数组为例。例如定义一个3行4列的二维数组，各元素的值如图2-7所示。

1	2	3	4
5	6	7	8
9	10	11	12

图2-7　二维数组

（1）定义时赋值

```
int[,] myArray=new int[3,4] {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

（2）使用赋值语句

```
int[,] myArray=new int[3,4];
myArray[0,0]=1;
myArray[0,1]=2;
…
myArray[2,3]=12;
```

（3）使用二重循环赋值

```
int[,] myArray=new int[3, 4];
for(int i=0;i<myArray.GetLength(0);i++)
{
    for (int j=0; j<myArray.GetLength(1); j++)
    {
        myArray[i, j]=i * 4+j+1;
    }
}
```

其中，myArray.GetLength(0)为二维数组的行数；myArray.GetLength(1)为二维数组的列数。

3）多维数组的显示

以已赋值的二维数组myArray为例，可以采用二重循环进行显示：

```
for (int i=0; i<myArray.GetLength(0); i++)
{
    for (int j=0; j<myArray.GetLength(1); j++)
    {
        Console.Write("{0,-4}  ", myArray[i, j]);
    }
    Console.WriteLine();
}
```

2.1.6　C#语言面向对象程序设计

1. 面向对象程序设计概述

程序设计就是针对某一要处理的问题，按照特定的方法、使用某种计算机语言设计出解





决该问题的计算机指令序列。自第一台计算机诞生以来,程序设计方法与程序设计语言都在不断地发展。程序设计方法经历了面向机器、面向过程和面向对象的发展历程,程序设计语言也经历了和面向过程相对应的低级语言(机器语言和汇编语言)到和面向过程、面向对象相对应的高级语言的发展历程。

面向机器的程序设计方法是以特定的硬件系统为主体,以汇编语言为代表的低级语言进行程序的设计。设计的程序运行速度和效率都很高,但因其是由二进制代码或符号表示的,可读性能差,不利于程序的编写和维护;而且程序是为特定的硬件系统专门设计的,可移植性能差。

面向过程的程序设计方法是以具体的解题过程为研究和实现的主体,以C语言为代表的高级语言来进行程序的设计。在面向过程程序设计中,问题被看做一系列需要完成的任务,用相应的函数来完成这些任务,面向过程程序设计的优点是易于理解和掌握,这种逐步细化问题的设计方法和大多数人的思维方式比较接近。然而,面向过程程序的控制流程由程序中的预定顺序来决定,这要求设计者在一开始就要对需要解决的问题有全面的了解。在问题比较复杂的时候,要做到这一点会比较困难,而且由于数据与处理方法是分开的,因此很可能产生数据和方法在结构上的不一致性。对程序运行起着重要作用的数据一般要做全局处理,若为了新的需求而改变某一数据结构,则对所有处理该数据的操作过程都要重新考虑,才能保证与数据的一致性。所以程序的修改及维护要花费大量的精力。

面向对象的程序设计方法是以需解决的问题中所涉及的各种对象为主体,将数据和对数据的操作封装在一起,作为一个整体来处理。程序的控制流程由运行时各种事件的实际发生来触发,而不再由预定顺序来决定,更符合实际需要。

在面向对象的设计中,不同类型的事或物都称为对象。对象具有各自的状态和行为,如某人是一个对象,具有姓名、身高、体重等状态和工作、学习等行为。具有共同的特性和行为的对象的抽象就是类。例如所有的人可以称为一个类。因此,类是对某一类对象的抽象,而对象是某一种类的实例。没有脱离对象的类,也没有不依赖于类的对象。

在面向对象程序开发人员的眼中,一切皆对象,如数据类型是对象,控件是对象,窗体是对象,ASP.NET页面也是对象。

2. 类的创建和使用

类是面向对象程序设计的基本构成模块。虽然在.NET Framework的类库中包含了大量的类供程序设计人员使用,但设计人员仍然需要针对特定问题的特定逻辑来定义自己的类。通过类可以定义数据元素以及定义对这些数据元素的操作。

1) 类的成员

类的成员主要包括常量、字段、方法、构造函数、属性、索引器、析构函数等。

类的成员又分为静态成员和实例成员。静态成员属于类,用static关键字修饰,使用“类名.静态成员名”方式访问;实例成员属于对象(类的实例),不用static关键字修饰,使用“对象名.实例成员名”方式访问。

2) 访问修饰符

访问修饰符用于确定类和类成员被访问的权限。类及类成员的访问修饰符及访问权限如表2-4所示。



表 2-4　类及类成员的访问修饰符及访问权限

类　别	访问修饰符	访问限制意义
类	public(公有)	可以在其他命名空间访问
	internal(内部)	只能在其所在命名空间访问
类成员	public(公有)	访问不受限制
	protected(受保护)	访问限于所在类和所在类的派生类
	internal(内部)	访问限于所在命名空间内
	private(私有)	访问限于所在类

访问修饰符可以省略。当类的访问修饰符省略时,默认是 internal 的;当类成员的访问修饰符省略时,默认是 private 的。

3) 类的创建

(1) 类的定义

定义类的格式为:

```
[访问修饰符]  class  类名
{
    [访问修饰符]类成员
}
```

(2) 字段的定义

字段是与对象或类相关联的变量。

定义字段的格式为:

```
[访问修饰符]类型 字段名;
```

(3) 属性的定义

属性结合了字段和方法的多个方面。可借助于 get 和 set 访问器对字段的值进行读写,为类字段提供保护,避免字段在对象不知道的情况下被更改。属性充分体现了对象的封装性。

定义属性的格式为:

```
public 类型 属性名
{
    set
    {
        设置属性值
    }
    get
    {
        获取属性值
    }
}
```

(4) 方法的定义

方法是类中可以执行的操作。





定义方法的格式为：

```
[访问修饰符] 返回值类型 方法名([形参列表])
{
    方法体
}
```

(5) 构造函数的定义

构造函数是类对象的初始化方法。

定义构造函数的格式为：

```
public 构造函数名(参数表)
{
    函数体
}
```

说明：

① 构造函数的访问修饰符必须是 public。

② 构造函数名和类名相同。

③ 构造函数没有返回类型和返回值。

④ 构造函数用于实例化对象，默认情况下定义类时会自动添加一个不可见的无参构造函数，用于实例化对象。

⑤ 可以添加有参数的构造函数，用于在实例化对象时给属性成员赋初值。但添加有参构造函数后，类中不会再自动添加无参构造函数，需要显式定义。

例 2-14 在项目中创建一个 Person 类，其中包括字段成员 name、age，属性成员 Name、Age，自我介绍的方法成员 SayHi()，无参构造函数及有参构造函数。

(1) 在解决方案资源管理器窗口右击项目名称，在弹出的快捷菜单中选择"添加"|"新建项"命令，打开"添加新项"对话框。如图 2-8 所示。



图 2-8 "添加新项"对话框

(2) 在"添加新项"对话框中，选择"类"选项，在"名称"文本框中输入类的名称(默认为 Class1)。单击"添加"按钮，在解决方案资源管理器中即可看到添加的类 Person.cs，如图 2-9





图 2-9　添加的类 Person. cs

所示。

（3）双击 Person. cs 类，在左侧窗格输入类的代码：

```
class Person                                            //类
{
    public Person(){}                                   //无参构造函数
    public Person(string name,int age)                  //有参构造函数
    {
        this.Name=name;
        this.Age=age;
    }
    private string name;                                //字段
    private int age;                                    //字段

    public string Name                                  //属性
    {
        get {return name;}
        set {name=value;}
    }
    public int Age                                      //属性
    {
        get {return age;}
        set {age=value;}
    }
    public void SayHi()                                 //方法
    {
        Console.WriteLine("我是{0},今年{1}岁了.", name, age);
    }

}
```

4）类的使用

类的使用主要是指调用类中的方法来实现某种功能。静态方法可直接通过“类名. 方法名”来调用；实例方法需要实例化一个对象，然后通过“对象名. 方法名”来调用。



实例化对象的格式为：

```
类名 对象名=new 类名([参数]);
```

其中，“类名([参数])”为类的构造函数，也即通过使用 new 操作符调用构造函数，实例化一个对象。[参数]为可选项，既可以是无参构造函数，也可以是有参构造函数。两者的区别是：使用无参构造函数实例化对象，需要通过“对象名.属性名”来给各个属性赋初值；而使用有参构造函数实例化对象，则可通过参数给属性赋初值。

例 2-15 调用 Person 类中的 SayHi()方法，在屏幕上显示个人信息。

在例 2-14 中，Person 类中的 SayHi()方法为实例方法，可以使用类中的无参构造函数 Person()或有参构造函数 Person(string name,int age)来实例化一个 Person 类的对象。

(1) 使用无参构造函数实例化对象

在项目的 Program 类的 Main()方法中输入下列代码。

```
Person person=new Person();              //实例化一个对象
person.Name="Jon";                       //通过属性 Name 给字段 name 赋值
person.Age=19;                           //通过属性 Age 给字段 age 赋值
person.SayHi();                          //调用 SayHi()方法
```

(2) 使用有参构造函数实例化对象

代码改为：

```
Person person=new Person("Jon",19);
person.SayHi();
```

运行程序后，会显示同样的效果，如图 2-10 所示。



图 2-10 创建的空白项目

3. 继承

继承是面向对象程序设计语言的一个重要特性。一个类可以从另一个类继承得到，被继承的类称为基类(或父类)，通过继承产生的新类称为派生类(或子类)。派生类继承了基类除构造函数外的所有成员，还可以根据处理问题的需要增加一些具有个性的新成员。

定义派生类的格式如下：

```
class 派生类名:基类名
{
    派生类代码
}
```

继承的过程，就是从一般到特殊的过程，通过继承可以实现代码的重用。

C#中的继承具有以下特点。

(1) 只允许单继承。即派生类不允许有多个基类。

(2) 继承具有传递性。如果类 A 派生类 B，类 B 又派生类 C，那么类 C 不仅继承类 B 的成员，也继承类 A 的成员。

(3) 派生类可以增加新成员，但不能移除继承的成员。

例 2-16 通过继承 Person 类派生一个 Student 类，增加字段 grade、属性 Grade 和方法



Test()三个成员。

(1) 在项目中新建类 Student，输入下列代码：

```
class Student:Person                                    //继承自 Person 类
{
    private int grade;                                  //添加的字段
    public int Grade                                    //添加的属性
    {
        get {return grade;}
        set {grade=value;}
    }
    public void Test()                                  //添加方法
    {
        Console.WriteLine("正在考试");
    }
}
```

(2) 在 Program 类的 Main()方法中实例化 Student 对象并赋值，调用继承的和添加的方法。

```
static void Main(string[] args)
{
    Student stu=new Student();                          //实例化对象
    stu.Name="Zhang";                                   //给继承的成员赋值
    stu.Age=20;
    stu.Grade=1;                                        //给添加的成员赋值
    stu.SayHi();                                        //调用继承的方法
    Console.WriteLine("{0}年级",stu.Grade);             //调用添加的成员
    stu.Test();                                         //调用添加的方法
}
```

(3) 运行程序，结果如图 2-11 所示。



图 2-11 类的继承

2.1.7 常用的类和函数

1. 命名空间和类

.NET 框架提供了功能强大的类库，类库中包含3000 多个类。为了能方便地使用类库中的类，在.NET 框架中使用命名空间来组织这些类。命名空间是类的逻辑分组，它组织成一个层次结构——逻辑树，这个树的根是 System。就像文件夹中可以包含文件和文件夹一样，命名空间中可以包含类和命名空间，所以.NET 框架中所有命名空间的集合是一个树状结构。

在使用类库中的类时，代码中要完整地体现类所在的命名空间和类名。如实例化一个连接 SQL Server 数据库类的对象，代码为：

```
System.Data.SqlClient.SqlConnection conn =  new System.Data.SqlClient.SqlConnection();
```

其中，SqlConnection 为类名，System.Data.SqlClient 为类所在的命名空间。这样的代码比较长，且有些语句可能要重复多次，使代码的书写量过大，而且给程序的维护带来了不便。为



了克服这方面的缺陷，在 C# 语言中使用 using 关键字来引入命名空间。如在类的开头添加 using System.Data.SqlClient 语句引入 System.Data.SqlClient 命名空间，这样在使用该命名空间中的类时就不用再写命名空间的名字了。如：SqlConnection conn=new SqlConnection()。

在使用 Visual Studio 2005 创建不同应用程序的类或窗体时，会自动引入常用的命名空间。如在 ASP.NET 应用程序中添加 Web 窗体时，自动添加下列代码。

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
```

这些命名空间中包含的类及其用途如表 2-5 所示。

表 2-5　命名空间中包含的类及其用途

命 名 空 间	包含的类及用途
System	包含用于定义类型、数组、字符串、事件、异常处理、接口、数据类型转换、数学计算、应用程序环境管理等的基础类
System.Data	包含组成 ADO.NET 体系结构的类
System.Configuration	包含用于处理配置数据的编程模型的类
System.Collections	包含用于管理对象集合的类
System.Web	包含用于实施浏览器、服务器通信和其他 Web 相关功能的类
System.Web.Security	包含用于在 Web 服务器应用程序中实现 ASP.NET 安全性的类
System.Web.UI	包含用于创建用户界面元素的 ASP.NET 服务器控件和页的类与接口
System.Web.UI.WebControls	包含用于在网页上创建 Web 服务器控件的类
System.Web.UI.WebControls.WebParts	包含用于创建可由最终用户修改(个性化设置)其外观和行为的网页的类与接口
System.Web.UI.HtmlControls	包含用于在 Web 窗体页上创建 HTML 服务器控件的类

.NET 框架类库的命名空间有 170 多个，另外在程序开发过程中可以定义自己的命名空间。需要时可以使用 using 语句引入这些命名空间。

2. Math 类

Math 类属于 System 命名空间，类中封装了三角函数、对数函数等其他各种常用的数学函数，也称为方法。这些方法全部为静态方法，可通过类名直接调用，其格式为：



```
Math.函数名(参数列表);
```

例如,Math 类中的 Abs()方法用于取得数值的绝对值,可通过 Math. Abs(—23.36)来计算数值—23.36 的绝对值。

Math 类中常用的数学运算方法如表 2-6 所示。

表 2-6　Math 类中常用的数学运算方法

函数名	用　　途	举　　例	结　果
Abs()	返回指定数字的绝对值	Math. Abs(—23.36)	23.36
Exp()	返回 e 的指定次幂	Math. Exp(1)	2.7182818
Floor()	返回舍去小数后的结果	Math. Floor(2.68)	2
Log10()	返回指定数字以 10 为底的对数	Math. Log10(100)	2
Log()	返回指定数字的对数(自然对数)	Math. Log(2.715)	0.996948
Max()	返回两个指定数字中较大的一个	Math. Max(3,5)	5
Min()	返回两个数字中较小的一个	Math. Min(3,5)	3
Pow()	返回指定数字的指定次幂	Math. Pow(2,3)	8
Round()	对浮点数进行四舍五入	Math. Round(2.68)	3
Sign()	返回表示数字符号的值。>0 为 1;<0 为—1;=0 为 0	Math. Sign(—34)	—1
Sqrt()	返回指定数字的平方根	Math. Sqrt(9)	3

3. String 类

String 类属于 System 命名空间,类中封装了对字符串进行操作的各种静态方法和非静态方法。例如. Compare()为静态方法,String. Compare(s1,s2)用于比较 s1 和 s2 两个字符串的大小关系;而. Equals()为静态方法,需要实例化一个 String 类的对象来调用,例如:

```
String s1="abcDbe"                    //实例化 String 对象 s1
s1.Equals("adde");                    //比较字符串"adde"和 s1 是否相等
返回 false
```

String 类中常用的字符串处理方法如表 2-7 所示。

例 2-17　编写控制台应用程序,实现以下功能。

(1) 提示用户输入邮箱名;

(2) 显示用户输入的邮箱名;

(3) 根据邮箱名中是否有“@”符号,显示用户名或提示邮箱名错误;

(4) 提示用户是否继续(输入 yes/no);

(5) 如用户输入的字符串进行小写转换后为“yes”,循环执行以上四步,否则结束程序。





表 2-7　String 类中常用的字符串处理方法

方　　法	说　　明	举　　例
Equals(string value)	比较两个字符串是否相等，与"=="作用一样。返回 true 或 false	String s1="abcDbe" s1.Equals("adde"); 返回 false
Compare(string a,string b)	比较两个字符串的大小关系。a<b 返回 −1；a==b 返回 0；a>b 返回 1	String s2="adde"; String.Compare(s1,s2); 返回 −1
IndexOf(string value)	获取字符串 value 在已知字符串中第一个匹配项的索引，找到返回索引，未找到返回 −1	s1.IndexOf("b"); 返回 1
LastIndexOf(string value)	获取字符串 value 在已知字符串中最后一个匹配项的索引，找到返回索引，未找到返回 −1	s1.LastIndexOf("b"); 返回 4
Join(string separator,string[] value)	把字符串数组 value 中的每个字符串用指定的分隔符 separator 连接，返回连接后的字符串	string[] s=new string[3] {"abc","de","wxyz"}; String.Join("#",s) 返回"abc#de#wxyz"
Split(char separator)	用指定的分割符 separator 分割字符串，返回分割后形成的数组	String[]s=s1.Split('b'); 返回数组"a cD e" foreach(String a in s)...
SubString(int startIndex,int length)	从指定的位置 startIndex 开始检索长度为 length 的字符串	s1.Substring(2,3); 返回"cDb"
ToLower()	转换为小写形式	s1.ToLower(); 返回"abcdbe"
ToUpper()	转换为大写形式	s1.ToUpper(); 返回"ABCDBE"
Trim()	去掉字符串两端的空格	s1.Trim(); 返回"abcDbe"

```
string choice;
do
{
    Console.WriteLine("请输入你的邮箱：");
    string email=Console.ReadLine();
    Console.WriteLine("你的邮箱是{0}",email);
    int position=email.IndexOf("@");
    if (position>0)
    {
        string name=email.Substring(0,position);
        Console.WriteLine("你的用户名是{0}",name);
    }
    else
    {
        Console.WriteLine("邮箱错误!");
    }
    Console.WriteLine("是否继续？(yes/no)");
    choice=Console.ReadLine().ToLower().Trim();
}
while(choice.Equals("yes"));
```



2.2 任务实施

2.2.1 为在线考试系统创建实体类

在本书中，将创建一个三层架构的 ASP. NET 应用程序，实现在线考试的功能。在线考试系统中，需要为数据库 OnLineExam1 中包含的每个表创建对应的实体类。

实体类是现实实体在计算机中的表示。它贯穿于整个架构，担负着在各层次及模块间传递数据的职责。在系统中，需要把数据库中的表映射为实体类，即把每一个表对应一个类，表中的字段对应类的字段和属性。

下面以表 exam_admin 为例，说明创建实体类的方法。

表 exam_admin 中的字段有 admin_id、admin_name、admin_pass、mgquestions、mgStudents 和 mgSystem，如图 2-12 所示。

表 - dbo.exam_admin　摘要

列名	数据类型	允许空
admin_id	int	☐
admin_name	varchar(20)	☐
admin_pass	varchar(20)	☐
mgquestions	int	☑
mgStudents	int	☑
mgSystem	int	☑
		☐

图 2-12　表 exam_admin 中的字段

SQL Server 2005 中的 int 数据类型对应. NET Framework 中的 int32，而 char、varchar、nchar、nvarchar 等对应 string。在项目中添加 exam_admin 类，输入代码如下：

```
public class examAdmin
{
    private int admin_id;                          //字段
    private string admin_name=String.Empty;
    private string admin_pass=String.Empty;
    private int mgquestions;
    private int mgStudents;
    private int mgSystem;
    public examAdmin()                             //构造函数
    {
    }
    public int Admin_id                            //属性
    {
        get {return admin_id;}
        set {admin_id=value;}
    }
    public string Admin_name
    {
        get {return admin_name;}
        set {admin_name=value;}
    }
    public string Admin_pass
    {
        get {return admin_pass;}
        set {admin_pass=value;}
    }
```




```
        public int Mgquestions
        {
            get {return mgquestions;}
            set {mgquestions=value;}
        }
        public int MgStudents
        {
             get {return mgStudents;}
            set {mgStudents=value;}
        }
        public int MgSystem
        {
            get {return mgSystem;}
            set {mgSystem=value;}
        }
    }
```

这样就创建了和表 exam_admin 对应的实体类 exam_admin。采用同样的方法可以创建与其他表对应的实体类。

小贴士

在字段、属性比较多的情况下，类中的代码量也比较大，为了方便阅览，可以使用#region、#endregion 指令把一段代码标记为有给定名称的一个块。如在 exam_admin 实体类中，选中属性部分的代码后右击，选择外侧代码、# region，输入名称“public Properties”，则代码折叠为一行，显示 public Properties，并且在左端有一个“+”号。当鼠标指针移到该行时，显示该语句块中的代码。也可单击“+”号展开查看。

2.2.2　在线考试系统倒计时的实现

当考生登录成功后，进入考试系统的 startExam. aspx 页面，并开始倒计时，在状态栏上显示本场考试总的时间及剩余时间，如图 2-13 所示。

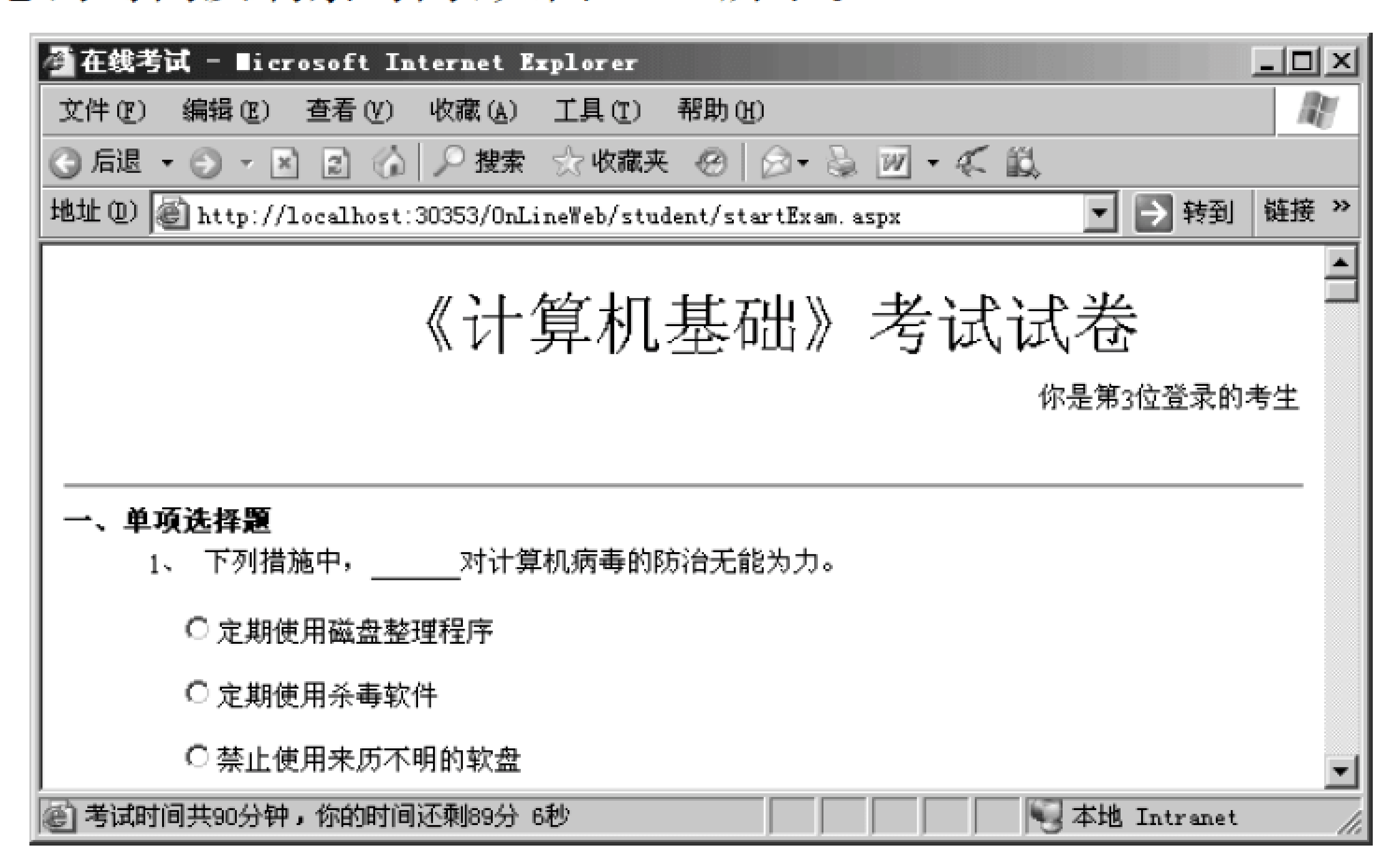


图 2-13　倒计时



倒计时的功能通过在页面中添加JavaScript来实现。JavaScript是一种脚本语言，它采用小程序段的方式实现编程，可以在浏览器中直接运行，无须服务器的支持。JavaScript与C#语言在函数和控制语句等方面有很多相似的地方。最大的区别是C#语言的功能更强大，语法规则更为严格和复杂。学习了C#语言或其他的高级语言，很容易看懂JavaScript代码。

新建一个ASP.NET网站，在Default.aspx页面中显示如下代码。

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs"
Inherits="Default2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    </div>
    </form>
</body>
</html>
```

在<head runat="server">...</head>代码块中，<title>无标题页</title>后添加如下脚本代码。

```
<link href="~/CSS/main.css" rel="stylesheet" type="text/css"/>
<script language="javascript">
    var timer=setInterval("CountTime()",1000);       //每1000毫秒调用一次计时函数
    var T_minute=90;                                 //设置考试的时间(分钟数)
    var T_second=0;                                  //设置秒数
      function CountTime()                           //计时函数
      {
          if (T_second==0)                           //如果秒数为0
          {
            if (T_minute==0)                         //如果分钟数为0
            {
                document.Form1.submitTry.click();    //交卷
            }
            else                                     //如果分钟数不为0
            {
                T_minute--;                          //分钟数减1
                T_second=59;                         //秒数设置为59
            }
          }
          else                                       //如果秒数不为0
          {
              T_second--;                            //秒数减1
          }
          window.status="考试时间共90分钟,你的时间还剩"+T_minute+"分 "+T_second+"秒";
                                                     //在状态栏上显示剩余时间
```




```
        }
    </script>
```

在代码中定义了计时函数 CountTime()，并通过 setInterval()函数每隔 1 秒(1000 毫秒)调用一次计时函数 CountTime()。在定义的计时函数 CountTime()中用到了选择语句的嵌套，在外层 if 语句中判断剩余时间的秒数是否为 0，如不为 0，秒数减 1；如为 0 则在内层 if 语句中判断剩余时间的分钟数。如分钟数不为 0 则减 1，秒数从 59 开始继续减少；如分钟数也为 0 则考试结束。

运行程序后，可在页面的状态栏中动态显示剩余的时间，如图 2-14 所示。

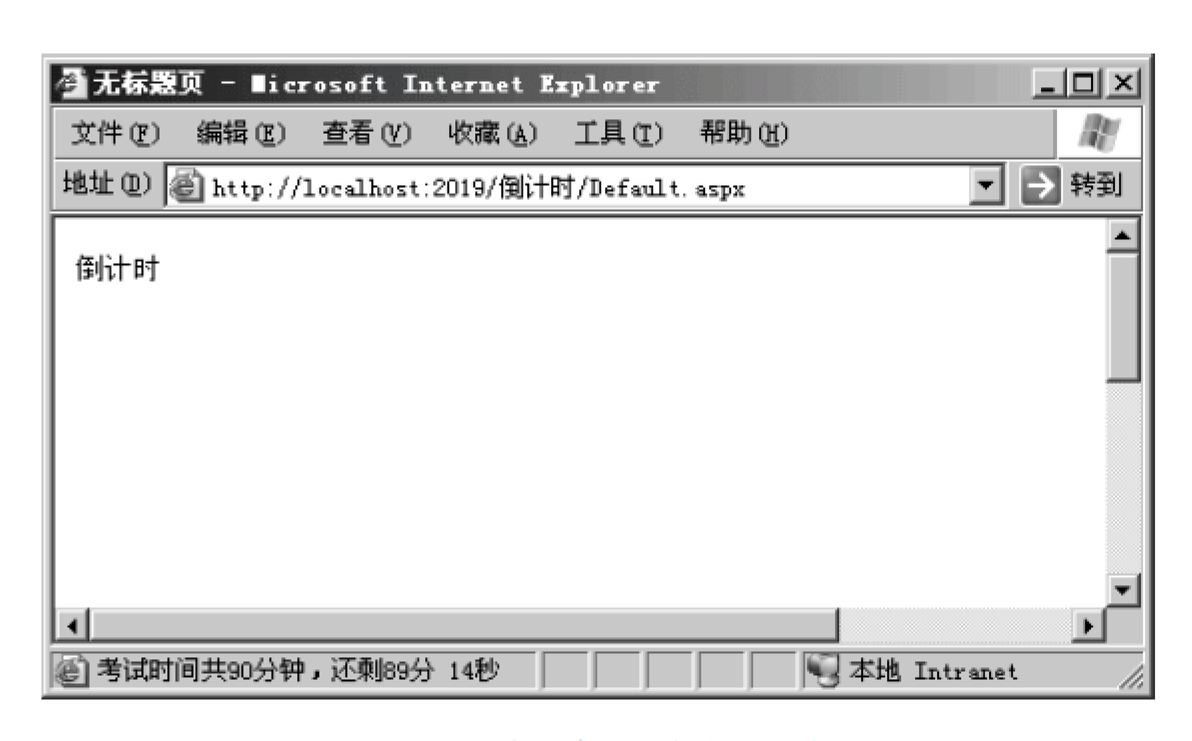


图 2-14　倒计时功能的实现

为简化问题的处理，在代码中把考试的时间 T_minute 设置为常数 90(分钟)，在实际应用时可以设置为一个变量，变量的值等于登录时选择的科目的考试时间。而每科目的考试时间在添加试卷时设定。可以实例化试卷管理类 examPapermg 的对象 paperMessage，然后通过 paperMessage.Test_time 获得考试的时间。

练　　习

1. 单项选择题

(1) 下面________是合法的变量名。

A. accp5.0　　B. CSharp　　C. 995　　D. Main

(2) 若要使用 SubString()方法从字符串 Superman 中截取 man 这个字符串，那么方法的两个参数应为________。

A. 5,3　　B. 5,7　　C. 6,3　　D. 6,8

(3) 声明了一个数组 Array[13]，则 Array[3]表示第________个元素。

A. 3　　B. 4　　C. 5　　D. 无法知道

(4) C#中的数据类型有值类型和________两种。

A. 结构类型　　B. 调用类型　　C. 引用类型　　D. 关系类型

(5) 下列________不是类的成员的访问修饰符。

A. private　　B. base　　C. public　　D. protected

2. 简答题

(1) 在 C#语言中，实现循环的语句有几种？



（2）什么是继承？它在软件设计中有什么作用？

实　　训

实训目的

（1）熟练掌握 C＃语言控制语句的使用。
（2）熟练掌握控制台应用程序的创建、编译和运行。
（3）熟练掌握类的创建、继承和使用。

实训内容

（1）创建控制台应用程序，实现用户登录功能。假设用户名为“abc”，密码为“123”，首先提示用户输入用户名，如果输入错误，提示“用户名错误”，退出程序；如果用户名正确，提示输入密码，如果输入错误，提示“密码错误！”，退出程序；如果密码正确，提示“登录成功！”。

（2）在上题的代码外加 do...while 循环，允许用户输入错误三次。

（3）创建一个电话类（Phone），其成员如下。

字段：品牌（brand）、价格（price）。

属性：品牌（Brand）、价格（Price）。

方法：打电话（Call()），在屏幕上显示“打电话”；响铃（Ringing()），在屏幕上显示“响铃”。

构造函数：Phone（string brand，int price），给字段赋初值。

实训指导

（1）创建控制台应用程序，在 Main()方法直接输入代码，如图 2-15 所示。

```
static void Main(string[] args)
{
    Console.Write("请输入用户名：");
    string name=Console.ReadLine();
    if (name=="abc")
    {
        Console.Write("请输入密码：");
        string pwd=Console.ReadLine();
        if (pwd=="123")
        {
            Console.WriteLine("登录成功!");
        }
        else
        {
            Console.WriteLine("密码错误");
        }
    }
    else
    {
        Console.WriteLine("用户名错误!");
    }
}
```

图 2-15　用户登录(一)



(2) 需要注意的问题：①循环次数变量 n 在循环外定义，每循环一次 n 在循环内加 1；②当用户登录成功，而且输入的错误次数小于 3 时，不应再提示用户输入用户信息。可通过在 Console. WriteLine("登录成功!")语句后加 return 语句跳出循环实现；③当输入的用户名正确、密码错误时，可能还会提示输入用户名，可以在 Console. Write("请输入用户名：")和 name=Console. ReadLine()语句外加 if(name !="abc"){}语句来判断。代码如图 2-16 所示。

(3) 在 Phone 类中定义有参构造函数 public Phone(string brand,int price)时，一般也定义一个无参构造函数 public Phone(){}。因为此类将作为父类使用，如果子类的构造函数没有使用 base 关键字指明调用父类的哪个构造函数，会隐式调用父类的无参构造函数。代码如图 2-17 所示。

```
int n=1;
string name="";
do
{
    if (name !="abc")
    {
        Console.Write("请输入用户名：");
        name=Console.ReadLine();
    }
    if (name=="abc")
    {
        Console.Write("请输入密码：");
        string pwd=Console.ReadLine();
        if (pwd=="123")
        {
            Console.WriteLine("登录成功!");
            return;
        }
        else
        {
            Console.WriteLine("密码错误!");
        }
    }
    else
    {
        Console.WriteLine("用户名错误!");
    }
    n++;
}while(n<4);
```

图 2-16 用户登录(二)

```
class Phone
{
    public Phone() {}
    public Phone(string brand, int price)
    {
        this.brand=brand;
        this.price=price;
    }
    private string brand;
    private int price;
    public string Brand
    {
        get {return brand;}
        set {brand=value;}
    }
    public int Price
    {
        get {return price;}
        set {price=value;}
    }
    public void Call()
    {
        Console.WriteLine("打电话");
    }
    public void Ranning()
    {
        Console.WriteLine("响铃");
    }
}
```

图 2-17 Phone 类



任务3　为在线考试系统制作导航系统

技能目标

能使用 TreeView 控件为在线考试系统制作树型目录，能使用 Menu 控件为在线考试系统制作动态菜单，能使用 SiteMapPath 控件为在线考试系统制作站点导航。

知识目标

掌握 TreeView 控件的用法，掌握 Menu 控件的用法，掌握 SiteMapPath 控件的用法；会使用各种导航控件建立不同的导航。

任务描述

建立网站的导航系统是 Web 开发中很重要的一个部分，要建立导航系统必须掌握 ASP.NET 提供的导航控件以及它们的功能和用法，并根据具体的情况使用不同的导航控件建立不同的导航方式。

本章任务如下：

◆ 使用 TreeView 控件为在线考试系统制作树型目录；

◆ 使用 SiteMapPath 控件为在线考试系统制作站点导航。

3.1　知识准备

3.1.1　XML 文件

XML(eXtensible Markup Language)即可扩展标记语言，是 Web 上的数据通用语言。它与 HTML 一样，都是 SGML(Standard Generalized Markup Language，标准通用标记语言)。XML 是 Internet 环境中跨平台的、依赖于内容的技术，是当前处理结构化文档信息的有力工具。它使开发人员能够将结构化数据从许多不同的应用程序传递到桌面，进行本地计算和演示，并且它还支持用户自己定义自己的一组标签，用来增强数据的自描述。XML 允许为特定应用程序创建唯一的数据格式。它还是在服务器之间传输结构化数据的理想格式。XML 文档是基于文本格式的，允许开发人员描述结构化数据并在各种应用之间发送和交换这些数据。

XML 实际上是 Web 上表示结构化信息的一种标准文本格式，它没有复杂的语法和包罗万象的数据定义。XML 的特点可以归纳为三点：先进特性、灵活性和自描述性。

(1) 先进特性

XML 继承了 SGML 的许多特性，首先是可扩展性。XML 允许使用者创建和使用他们自己的标记而不是 HTML 的有限词汇表。这一点至关重要，企业可以用 XML 为电子商务和供应链集成等应用定义自己的标记语言，甚至特定行业可以一起来定义该领域的特殊标记语言，作为该领域信息共享与数据交换的基础。

(2) 灵活性

HTML 很难进一步发展，就是因为它是格式、超文本和图形用户界面语义的混合，要同时发展这些混合在一起的功能是很困难的。而 XML 提供了一种结构化的数据表示方式，使得用户界面分离于结构化数据。所以，Web 用户所追求的许多先进功能在 XML 环境下更容易实现。

(3) 自描述性

XML 文档通常包含一个文档类型声明，因而 XML 文档是自描述的。不仅人能读懂 XML 文档，计算机也能处理。XML 表示数据的方式真正做到了独立于应用系统，并且数据能够重用。XML 文档被看做是文档的数据库化和数据的文档化。

除了上述先进特性以外，XML 还具有简明性。它只有 SGML 约 20%的复杂性，但却具有 SGML 功能的约 80%。XML 比完整的 SGML 简单得多，易学、易用并且易实现。另外，XML 也吸收了人们多年来在 Web 上使用 HTML 的经验。XML 支持世界上几乎所有的主要语言，并且不同语言的文本可以在同一文档中混合使用，应用 XML 的软件能处理这些语言的任何组合。所有这一切将使 XML 成为数据表示的一个开放标准，这种数据表示独立于机器平台、供应商以及编程语言。它将为网络计算注入新的活力，并为信息技术带来新的机遇。目前，许多大公司和开发人员已经开始使用 XML，包括 B2B 在内的许多优秀应用已经证实了 XML 将会改变今后创建应用程序的方式。

由于 XML 具有如此多的优点，因此在 ASP.NET 中很多文件都采用了 XML 的存储方式，如：Web.config(站点配置文件)、SiteMap(站点地图)等。

下面通过一个小例子来了解 XML 的文档结构。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <bookstore>
    <book catalog="Programming">
        <title>C#</title>
        <author>Bjarne Stroustrup</author>
        <year>2009</year>
    </book>

    <book catalog="Networking">
        <title>TCP/IP</title>
        <author>Richard Stevens</author>
        <year>2009</year>
    </book>
  </bookstore>
```

第 1 行便是 XML 声明，标识使用的 XML 版本号。

第 2 行是根元素。所有的 XML 文件必须有并且只能有一个根元素，它是文件中最外面的标签。所有其他的标签必须包含在这个标签之内来组成一个有效的 XML 文件。

第 3～7 行是根元素的一个节点，第 8～12 行是根元素的另一个节点。



第 13 行是根元素的结束标志。

注意：XML 文件区分大小写。

3.1.2 TreeView 控件

树型导航是网站中广泛采用的一种导航方式，利用树型导航，可以很容易地了解到文件和目录之间的层次结构。如：程序员大本营的论坛采用的就是这种形式，如图 3-1 所示。



图 3-1 树型导航示例

在 ASP. NET 中，服务器控件 TreeView 用于以树型结构显示分层数据，如目录或文件。它支持以下功能。

(1) 自动数据绑定，该功能允许将控件的节点绑定到分层数据(如 XML 文档)。

(2) 通过与 SiteMapDataSource 控件集成提供对站点导航的支持。

(3) 可以显示为可选择文本或超链接的节点文本。

(4) 可通过主题、用户定义的图像和样式自定义外观。

(5) 通过编程访问 TreeView 对象模型，使用户可以动态地创建树、填充节点以及设置属性等。

(6) 通过客户端到服务器端的回调填充节点(在受支持的浏览器中)。

(7) 能够在每个节点旁边显示复选框。

根据数据类型的不同，TreeView 的使用方式有三种：手动添加数据项、使用站点地图作为数据源和使用 XML 文件作为数据源。

3.1.3 站点地图

站点地图是对站点结构的 XML 描述，它用来描述网站的逻辑结构。文件的后缀是 sitemap。TreeView 控件、Menu 控件和 SiteMapPath 控件都可以结合站点地图所确定的逻辑关系来给网站导航。

创建站点地图的方法如下：

右击网站，在弹出的快捷菜单中选择“添加新项”|“站点地图”命令，以打开站点地图文件。默认的站点地图的文件名为 Web. sitemap，通常放在站点根目录下。初次创建的站点地图文件的框架如下所示。

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
    <siteMapNode url="" title=""  description="">
        <siteMapNode url="" title=""  description=""/>
        <siteMapNode url="" title=""  description=""/>
    </siteMapNode>
</siteMap>
```

编写站点地图文件时，应注意以下几点。



(1) 一个站点地图有一个根节点(siteMap 元素)。

(2) siteMap 下一级只有一个 siteMapNode 节点,在该 siteMapNode 节点下可以包含很多个 siteMapNode 节点。

(3) 一个 siteMapNode 节点描述一个页面。

(4) 每个节点具有 3 种属性:节点标志、调用网页的 URL 以及内容提示等。

① url:该节点调用网页的 URL。

② title:节点标志。

③ description:说明性文本,作为智能提示的字符串。

(5) 在站点地图中,同一个 URL 只能出现一次。

3.1.4　SiteMapPath 控件

站点地图创建好之后,就可以利用 SiteMapPath 控件来显示浏览者当前的位置。SiteMapPath 控件也叫“面包屑”导航控件。该控件必须与站点地图文件相结合,而且最好放在主控页中。使用该控件时,不需要编写任何代码,只要在相应位置有已经写好的站点地图文件,并将 SiteMapPath 控件直接拖入窗体,它就会自动与站点地图文件相结合,形成导航效果。

SiteMapPath 控件只能显示从根节点到当前节点之间的路径,利用它只能返回某个页面,而不能向前选择页面。

在 SiteMapPath 控件中,除有一些通用的属性之外,还有以下一些特殊的属性可以用来改变控件的显示界面。

(1) PathSeparator 属性

PathSeparator 属性的功能为控制分隔符的样式;可以通过编辑模板更改分隔符为任意样式;分隔符模板为<PathSparatorTemplate>。

(2) PageLevelsDisplayed 属性

PageLevelsDisplayed 属性的功能为控制导航显示的级数;导航过深时,会影响美观,可通过该属性控制级数。

(3) PathDirection 属性

PathDirection 属性的功能为显示路径的方向,有两种选择。

① RootToCurrent:从根到当前页面。

② CurrentToRoot:从当前页面到根。

3.2　任务实施

3.2.1　为在线考试系统制作树型目录

为方便与考试相关的各种信息的管理,在线考试系统需要建立一个后台管理系统,采用控件 TreeView 用于以树型结构显示分层数据,效果如图 3-2 所示,在该系统中使用 XML 文件作为数据源来显示数据。具体实现的步骤如下。



1. 编写 XML 文件作为数据源

首先要编写一个 XML 文件,作为数据源。

如图 3-3 所示,在解决方案资源管理器中,新建文件夹 admin,后台管理的所有页面都放在 admin 文件夹中。右击 admin,在弹出的快捷菜单中选择“添加新项”命令,打开“添加新项”对话框,如图 3-4 所示。在图 3-4 中,选择“XML 文件”模板,在“名称”文本框中输入“admin_menu. xml”,然后单击“添加”按钮。

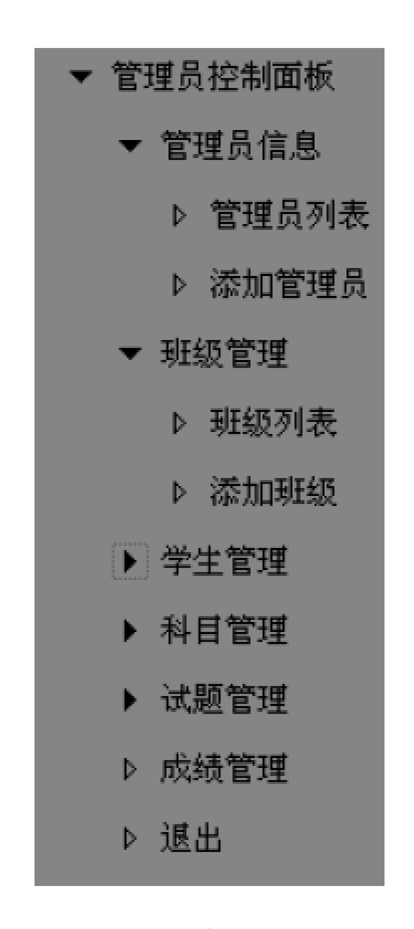


图 3-2 树型目录



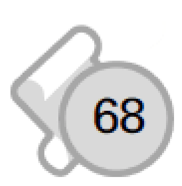
图 3-3 添加新项



图 3-4 新建 XML 文件

admin_menu. xml 文件的内容如下:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteRoot Id="root" url="" title="管理员控制面板" description="">
  <siteMapNode url="" title="管理员信息" description="">
    <siteMapNode url="~\Admin\AdminList.aspx" title="管理员列表" description=""/>
    <siteMapNode url="~\Admin\AdminDetails.aspx" title="添加管理员" description=""/>
  </siteMapNode>
```






```
    <siteMapNode url="" title="班级管理" description="">

      <siteMapNode url="~\Admin\ListClass.aspx" title="班级列表" description=""/>
      <siteMapNode url="~\Admin\AddClass.aspx" title="添加班级" description=""/>
    </siteMapNode>
    <siteMapNode url="" title="学生管理" description="">

      <siteMapNode url="~\Admin\ListAllStudents.aspx" title="学生列表" description=""/>
      <siteMapNode url="~\Admin\RegisterStudent.aspx" title="添加学生" description=""/>
    </siteMapNode>
    <siteMapNode url="" title="科目管理" description="">
      <siteMapNode url="~\Admin\ListSubject.aspx" title="科目列表" description=""/>
      <siteMapNode url="~\Admin\AddSubject.aspx" title="添加科目" description=""/>
      <siteMapNode url="~\Admin\SubjectExam.aspx" title="科目试卷设置" description=""/>
    </siteMapNode>
    <siteMapNode url="" title="试题管理" description="">

      <siteMapNode url="~\Admin\ListAllQuestions.aspx" title="试题列表" description=""/>
      <siteMapNode url="~\Admin\QuestionLists.aspx" title="试题列表" description=""/>
      <siteMapNode url="~\Admin\QuestionDetails.aspx" title="添加试题" description=""/>
    </siteMapNode>
    <siteMapNode url="~\Admin\stuScore.aspx" title="成绩管理" description="">
    </siteMapNode>
    <siteMapNode url="~\Admin\LoginOut.aspx" title="退出" description="管理员退出">

    </siteMapNode>
  </siteRoot>
```

该文件显示了后台管理系统所有文件的层次结构。

其中，siteRoot：最外层节点。siteMapNode：对应于页面的节点，一个节点描述一个页面。siteMapNode 可以嵌套。url：节点对应的页面地址。title：树型目录的显示文本。description：说明性文字，类似注释。

2. 为 TreeView 控件设置数据源

（1）在解决方案资源管理器中，右击 admin 文件夹，在弹出的快捷菜单中选择“添加新项”命令，在出现的“添加新项”对话框中选择“Web 窗体”模板，在“名称”文本框中输入“test1. aspx”，然后单击“添加”按钮。

（2）在新建页面中，切换到设计视图，从工具箱的“导航”栏拖入 TreeView 控件。下面为 TreeView 控件配置数据源。

（3）把光标指向 TreeView 控件，单击右上方的黑色三角按钮，显示“TreeView 任务”窗口，如图 3-5 所示。在“选择数据源”下拉列表框中选择“新建数据源”选项，打开“数据源配置向导”对话框，如图 3-6 所示。

（4）在图 3-6 所示的对话框中选择“XML 文件”选项，单击“确定”按钮，打开“配置数据源”对话框，如图 3-7 所示。

（5）单击“数据文件”文本框后的“浏览”按钮，然

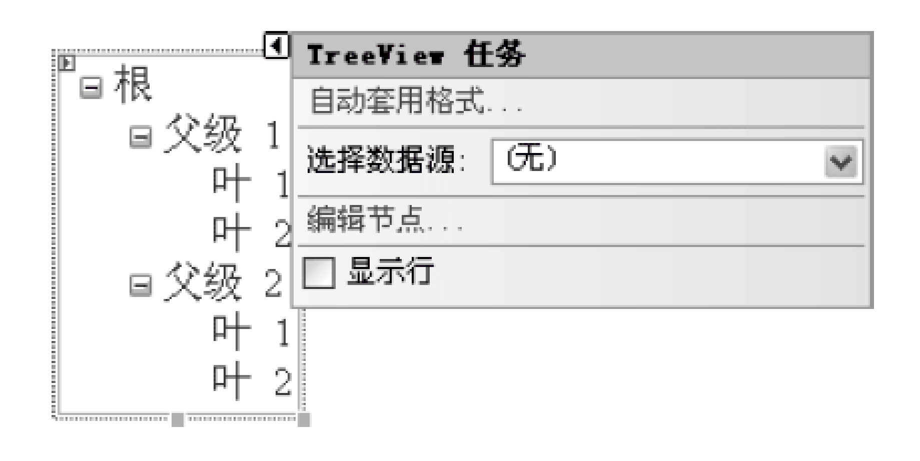


图 3-5　为 TreeView 控件配置数据源





图 3-6 “数据源配置向导”对话框



图 3-7 “配置数据源”对话框

后选择 admin_menu. xml 文件。

（6）单击“确定”按钮，即可完成 XML 文件的绑定，形成如图 3-8 所示的绑定效果。

3．编辑数据绑定

在刚才的操作中，绑定过 XML 文件后，TreeView 并未识别我们的意图。因此，需要进一步编辑数据绑定，使其可以正常显示 XML 文件中的内容。

将光标指向 TreeView 控件，单击右上方的黑色三角按钮，显示“TreeView 任务”窗口，在“选择数据源”下拉列表框中选择“编辑 TreeNode 数据绑定”选项，打开“TreeView DataBindings 编辑器”对话框，添加 siteRoot 节点和 siteMapNode 节点。将这两个节点的 TextField 绑定 title 属性，作为显示字段，将 NavigateUrlField 绑定 url 属性，作为链接字段，如图 3-9 所示。



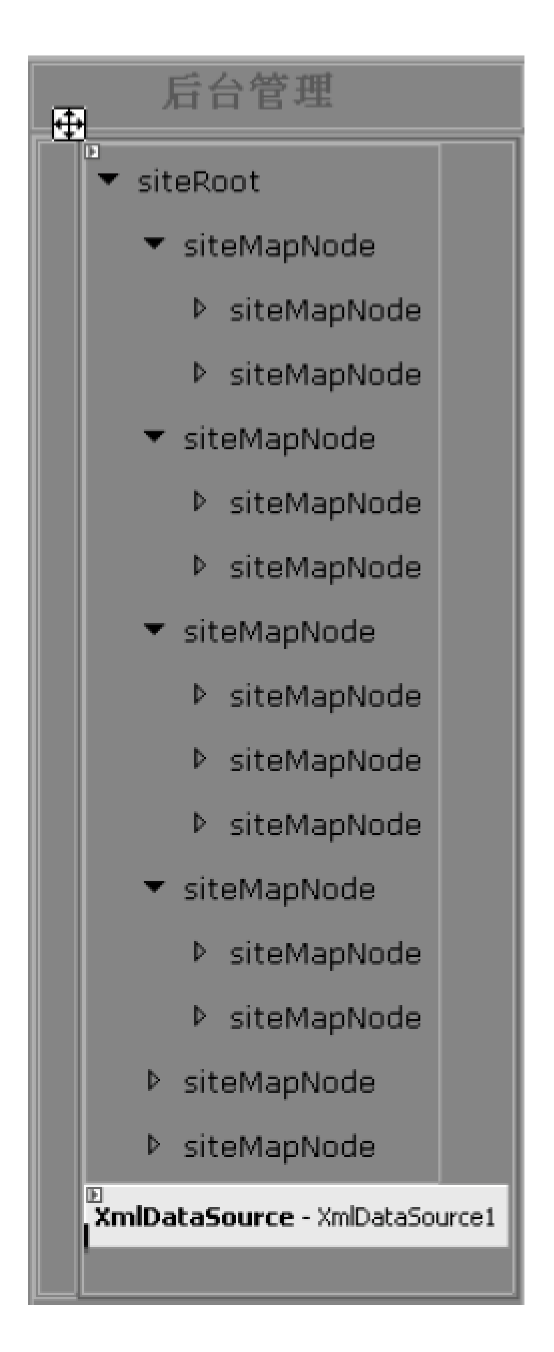


图 3-8　与 XML 文件绑定之后的 TreeView 控件



图 3-9　TreeView DataBindings 设置

4. 设置格式

这样，TreeView 控件就可以正常显示了。此时，还可以通过选择智能标记中的"自动套用格式"命令(图 3-10)，打开"自动套用格式"对话框，如图 3-11 所示，来给 TreeView 添加一种样式效果。这里选择"箭头"样式，设置好的效果如图 3-2 所示。



图 3-10　智能标记中选择"自动套用格式"



图 3-11　"自动套用格式"对话框



小贴士

TreeView 的数据源可以有很多种，除了 XML 文件外，还可以是站点地图；另外，TreeView 控件还可手动设置各个 TreeNode 的内容。

3.2.2 为在线考试系统制作站点导航

如果在系统的后台管理页面中显示出当前位置，如图 3-12 所示，则管理起来更加方便。下面为在线考试系统管理员后台制作站点导航，具体步骤如下。

您的位置：首页 >管理员后台 >学生管理 >学生列表

图 3-12 站点导航

1. 编写站点地图文件作为数据源

编写好的站点地图文件如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
    <siteMapNode url="~/Index.aspx" title="首页"  description="">
    <siteMapNode url="~\Admin\test1.aspx" title="测试页"  description="">
</siteMapNode>

    <siteMapNode Id="" url="" title="管理员后台" description="">
      <siteMapNode url="" title="管理员信息" description="">
        <siteMapNode url="~\Admin\AdminList.aspx" title="管理员列表" description=""/>
        <siteMapNode url="~\Admin\AdminDetails.aspx" title="管理员详细信息"
        description=""/>
      </siteMapNode>

      <siteMapNode url="" title="班级管理" description="">
        <siteMapNode url="~\Admin\ListClass.aspx" title="班级列表" description=""/>
        <siteMapNode url="~\Admin\AddClass.aspx" title="添加班级" description=""/>
      </siteMapNode>

      <siteMapNode url="" title="学生管理" description="">
        <siteMapNode url="~\Admin\ListAllStudents.aspx" title="学生列表"
        description=""/>
        <siteMapNode url="~\Admin\StudentDetails.aspx" title="学生详细信息"
        description=""/>
        <siteMapNode url="~\Admin\RegisterStudent.aspx" title="添加学生"
        description=""/>
      </siteMapNode>

      <siteMapNode url="" title="科目管理" description="">
        <siteMapNode url="~\Admin\ListSubject.aspx" title="科目列表" description=""/>
        <siteMapNode url="~\Admin\AddSubject.aspx" title="添加科目" description=""/>
        <siteMapNode url="~\Admin\SubjectExam.aspx" title="科目试卷设定"
```






```
      description=""/>
    </siteMapNode>

    <siteMapNode url="" title="试题管理" description="">
      <siteMapNode url="~\Admin\QuestionDetails.aspx" title="试题详细信息"
      description=""/>
      <siteMapNode url="~\Admin\ListAllQuestions.aspx" title="试题列表"
      description=""/>
      <siteMapNode url="~\Admin\QuestionLists.aspx" title="试题列表"
      description=""/>
    </siteMapNode>

    <siteMapNode url="~\Admin\stuScore.aspx" title="成绩管理" description="">
    </siteMapNode>

    <siteMapNode url="~\Admin\LoginOut.aspx" title="退出" description="管理员退出">
    </siteMapNode>
  </siteMapNode>
 </siteMapNode>
</siteMap>
```

2. 添加 SiteMapPath 控件

在“test1.aspx”中，拖入 SiteMapPath 控件。然后执行“test1.aspx”，没有看到导航信息，这是因为 SiteMapPath 控件要求站点地图中必须有当前页导航，否则该站点导航控件不会显示。

为此需要在站点地图中添加如下代码。

```
<siteMapNode url="~\Admin\test1.aspx" title="测试页"  description="">
</siteMapNode>
```

这样，单击相应的页面标题时，即可切换到相应的页面。

小贴士

SiteMapPath 控件并不需要指定数据源，它的数据源就是站点地图。

3.3　知识和技能扩展——Menu 控件与网站菜单

3.3.1　网站菜单

除了树型导航菜单之外，很多网站也采用类似于 Windows 系统的层级菜单的下拉菜单来实现站点导航。下拉菜单是网上最常见到的效果之一，这种菜单通常有静态显示和动态显示两部分，静态部分用于静态显示导航；动态部分是隐藏的，当用户需要的时候，用鼠标轻轻一点或是将光标移过去，就出现一个更加详细的菜单，它不仅节省了网页排版的空间，使网页布局简洁有序，而且一个新颖美观的下拉菜单，更是为网页增色不少。如图 3-13 就是



微软官网的一个下拉菜单的示例。

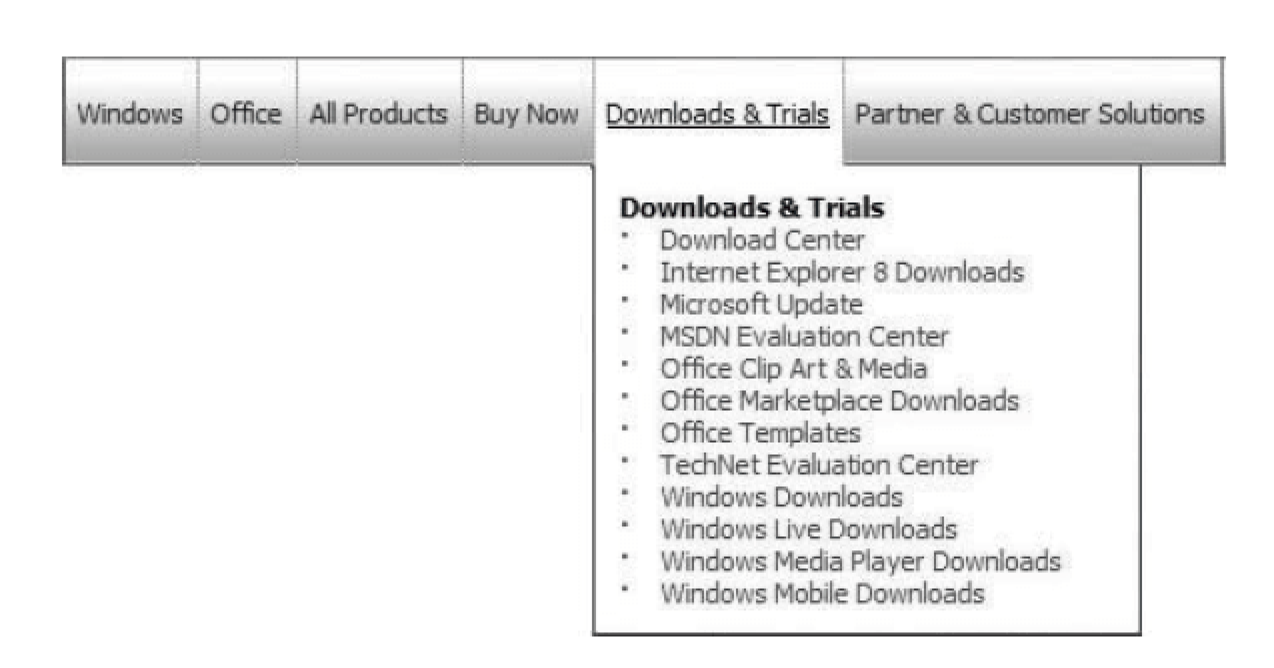


图 3-13　网站下拉菜单示例

3.3.2　Menu 控件

1. Menu 控件介绍

实现下拉菜单的方法有很多，比如手动编写 JavaScript 代码，或者利用 Dreamweaver 编写，但是这两种方法都需要有很长的代码才能实现动静结合的菜单效果。在 ASP. NET 中提供了一种 Menu 控件，利用 Menu 控件，无须编写代码，就可以开发 ASP. NET 网页的静态和动态显示菜单，可以在 Menu 控件中直接配置其内容，也可通过将该控件绑定到数据源的方式来指定其内容。此外，同样无须编写任何代码，便可控制 ASP. NET Menu 控件的外观、方向和内容。

Menu 控件是用来开发静态和动态显示菜单的控件，因此它提供了两种显示模式：静态模式和动态模式。静态显示意味着 Menu 控件始终是完全展开的，整个结构都是可视的，用户可以单击任何部位。这种方式类似于 TreeView 那样的显示方式，将所有节点罗列在页面上；在动态显示的菜单中，只有指定的部分是静态的，而对于动态部分，只有当用户将鼠标指针放置在父节点上时才会显示其子菜单项，移开鼠标时，子菜单项会自动消失。

2. Menu 控件的使用

Menu 控件的使用方法如下。

(1) 添加 Menu 控件

在页面上拖入 Menu 控件，如图 3-14 所示。此时由于 Menu 控件没有绑定任何数据，因此不会看到菜单导航的效果。

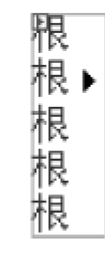


图 3-14　初始状态

(2) 编辑 Menu 控件

编辑 Menu 控件的方法有两种：手动编辑和绑定数据源。首先介绍手动编辑方式。在智能标记中选择“编辑菜单项”命令(图 3-15)，打开“菜单项编辑器”对话框，并在里面输入相应的菜单项名称，如图 3-16 所示。

然后介绍绑定数据源的方法。如果要通过绑定数据源来显示数据，可以选择图 3-15 智





图 3-15　选择“编辑菜单项”命令



图 3-16　编辑菜单项

能标记中“选择数据源”下拉列表框中的“新建数据源”选项，如图 3-17 所示，可打开如图 3-18 所示的“数据源配置向导”对话框。



图 3-17　选择新建数据源

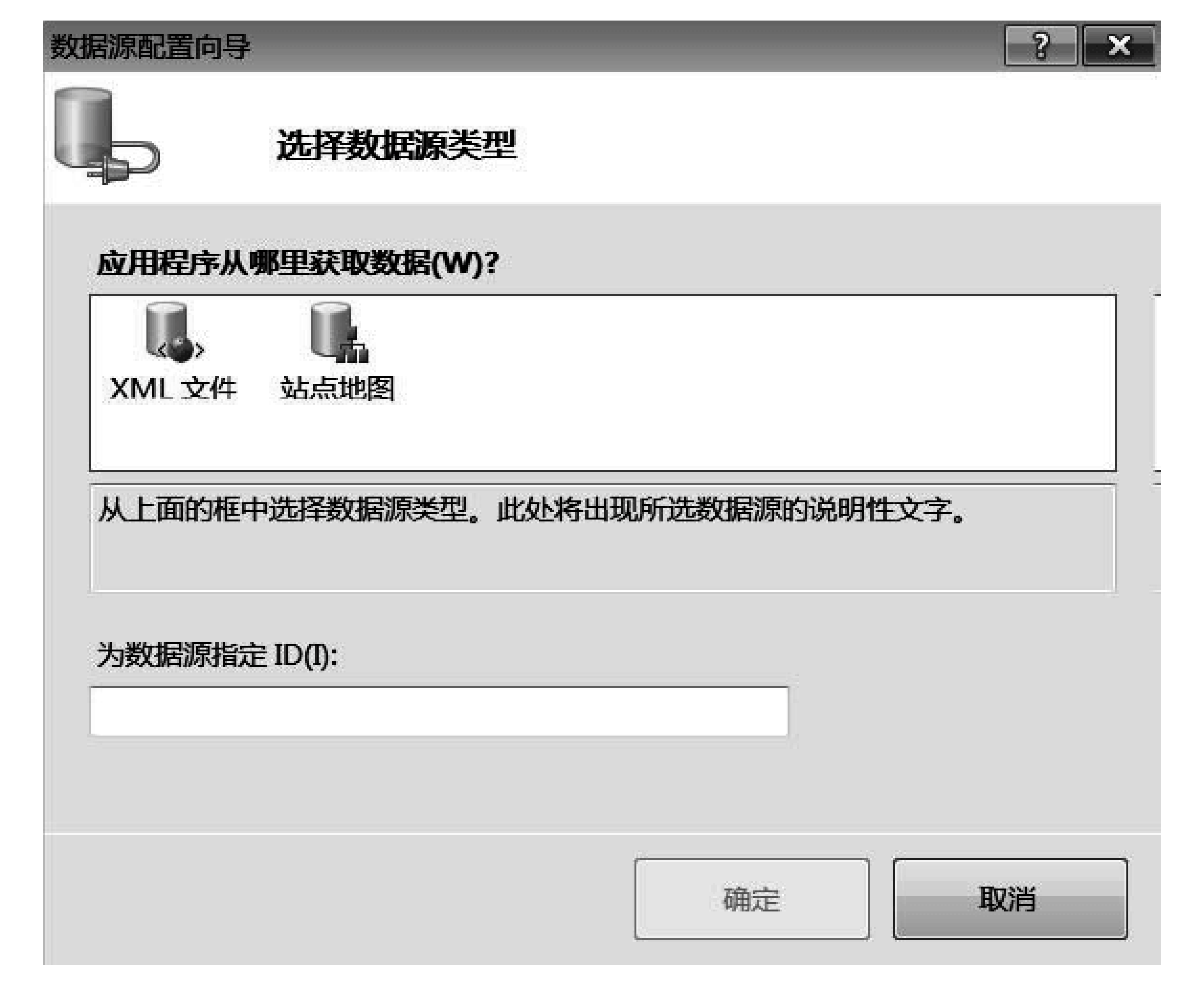


图 3-18　“数据源配置向导”对话框

在这个对话框中，可以选择“XML 文件”或者“站点地图”作为数据源。这里选择“站点地图”作为数据源，单击“确定”按钮之后，完成绑定，界面最终运行效果如图 3-19 所示。



图 3-19　Menu 控件最终效果



练　　习

1. 单项选择题

(1) 常用的导航控件不包括________。

A. TreeView 控件　B. GridView 控件　C. Menu 控件　D. SiteMapPath 控件

(2) 绑定过 XML 文件后，TreeView 控件通常无法正常显示，需要通过________节点来绑定 title 属性。

A. siteRoot　B. siteMapNode　C. TextField　D. NavigateUrlField

(3) 站点地图中，通过________来设置调用网页的 URL。

A. url　B. description　C. title　D. PostUrl

(4) ________属性控制 SiteMapPath 控件分隔符的样式。

A. PathSeparator　B. PageLevelsDisplayed

C. PathDirection　D. Path

2. 简答题

(1) 简述三种导航控件的异同点。

(2) 简述站点地图的基本结构。

实　　训

实训目的

(1) 熟练掌握 TreeView 控件的使用方法。

(2) 熟练掌握 Menu 控件的使用方法。

(3) 熟练掌握 SiteMapPath 控件的使用方法。

实训内容

(1) 按下面步骤完成实训。

① 启动 Visual Studio 2005。

② 打开 Visual Studio 开发环境中解决方案资源管理器、工具箱、属性窗口和设计视图。

③ 新建一个网站 Exercise 3，使用三个导航控件实现网站导航效果。

④ 将上面的网站发布。

⑤ 配置 IIS，在 IIS 服务器上浏览相应页面。

(2) 完成在线考试系统的导航设计。

实训指导

(1) 使用“视图”菜单，可打开解决方案资源管理器和属性窗口。将光标悬浮在工具箱



图标上面可显示工具箱。单击窗口下方"设计"按钮可切换到设计视图。

（2）完成如下所示的站点地图。

```
<?xml version="1.0" encoding="utf-8"?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode title="网上商城" description="" url="Index.aspx">
    <siteMapNode title="图书" url="Book.aspx" description=""/>
    <siteMapNode title="衣服" url="Coat.aspx" description=""/>
  </siteMapNode>
</siteMap>
```

（3）在网站中新建网页 Index.aspx，并在上面拖入一个 Menu 控件，并且让该控件的数据源为刚才建好的站点地图，运行之后可形成如图 3-20 所示的效果。

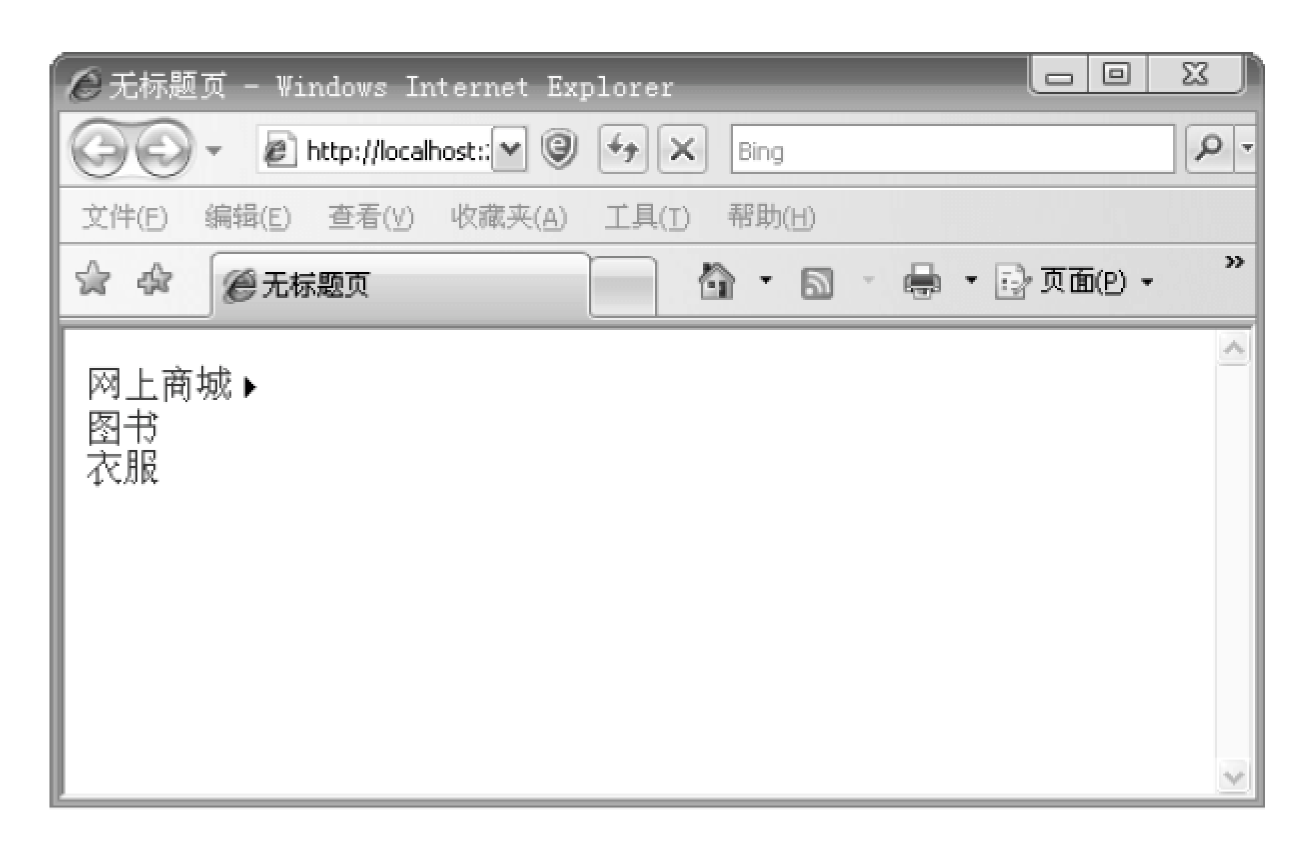


图 3-20　Index 页面

（4）在网站中新建网页 Book.aspx，并在上面拖入一个 SiteMapPath 控件和一个 TreeView 控件，并为 TreeView 控件手动添加导航菜单，运行之后可形成如图 3-21 所示的效果。



图 3-21　Book 页面

（5）仿照第(4)步，在网站中新建网页 Coat.aspx，并在上面拖入一个 SiteMapPath 控件和一个 TreeView 控件，并为 TreeView 控件手动添加导航菜单，运行之后可形成如图 3-22



所示的效果。



图 3-22　Coat 页面

任务4　三层架构与系统框架

技能目标

会搭建三层架构的系统框架，为模型层创建实体类。

知识目标

掌握三层架构的原理，掌握三层架构的创建与层之间的引用。

任务描述

使用三层架构进行系统开发的基础是要搭建系统框架，本章将进行"在线考试系统"系统框架的搭建。

本章任务如下：

- 用三层架构搭建"在线考试系统"系统框架；
- 为在线考试系统创建模型层实体类。

4.1　知识准备

4.1.1　三层架构介绍

传统的程序设计中，界面代码、业务逻辑代码以及操作数据库的代码是混合在一起的，设计人员必须对美工、业务逻辑和数据库各方面的知识都非常了解，如果要对程序的数据库、业务逻辑或界面的某一地方进行微小的改动，可能要涉及整个程序大面积的修改，给程序的开发和维护带来了极大的不便。

为了克服上述问题，人们总结程序开发的经验，提出采用分层的方式来进行处理，具体来讲，就是把不同功能的代码放到不同的项目，规定各项目之间的依赖关系和接口，分块进行开发和维护。分层的方案有很多种，其中影响力最大也最成熟的就是三层架构的分层方案。

通常意义上的三层架构的"三层"是指用户界面表示层（UI）、业务逻辑层（BLL）和数据访问层（DAL），如图 4-1 所示。

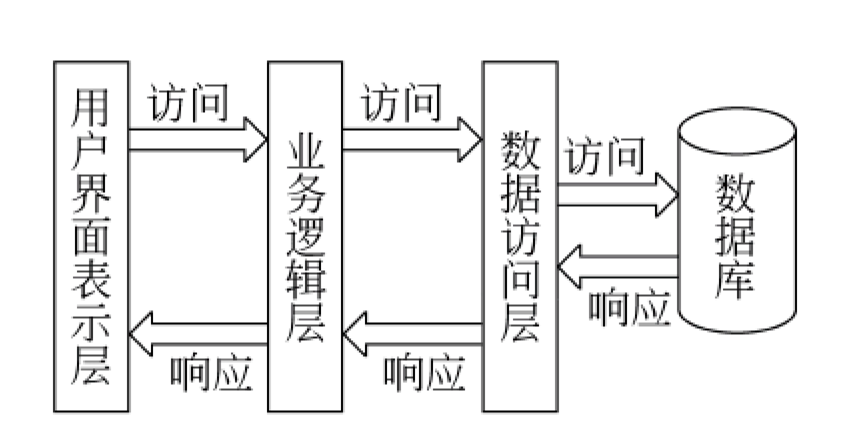


图 4-1　三层架构

（1）用户界面表示层：直接和用户进行交互，把数据内容呈现给用户或接收用户输入的数据的部分。对



于 Web 应用程序来说，表示层就是网页；对于 Windows 应用程序来说，表示层就是 Windows 窗体。

（2）业务逻辑层：主要是处理业务方面的逻辑，如判断用户输入的内容是否符合要求，负责把用户输入的数据传输给数据访问层，并把来自数据访问层的数据返回给用户。业务逻辑层是用户界面表示层和数据访问层之间通信的桥梁，它在体系架构中的位置很关键，处于数据访问层与用户界面表示层中间，起到了数据交换中承上启下的作用。

（3）数据访问层：主要是对原始数据（数据库或者文本文件等存放数据的形式）的操作层，只有它能够直接访问数据库，对数据库进行增、删、改、查的操作，其他的两个层都不能直接访问数据库。

在三层架构中，客户端不直接与数据库进行交互，而是经过中间层与数据库进行交互。

三层架构具有以下优点。

（1）开发人员可以只关注整个结构中的其中某一层；

（2）可以很容易地用新的实现来替换原有层次的实现；

（3）可以降低层与层之间的依赖；

（4）有利于标准化；

（5）利于各层逻辑的复用。

小贴士

三层架构中除了用户界面表示层、业务逻辑层和数据访问层之外，还有一个模型层。模型层包含所有与数据库中的表相对应的实体类。三层之间的数据传递是通过传输模型层的实体对象来实现的，如图 4-2 所示。

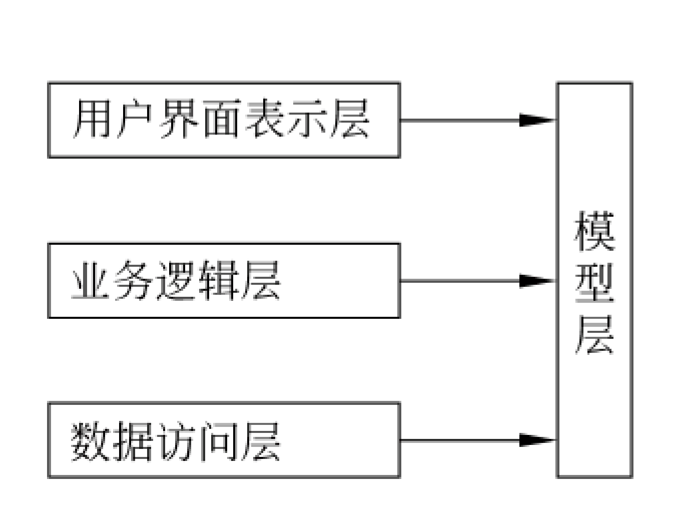


图 4-2　模型层与其他层的关系



图 4-3　在线考试系统框架结构

4.1.2　在线考试系统的系统结构

在线考试系统的架构如图 4-3 所示，先创建解决方案（取名 OnLine），在解决方案下创建 4 个项目：第一个项目是用户界面表示层（取名 OnLineWeb），第二个项目是业务逻辑层（取名 OnLineBLL），第三个项目是数据访问层（取名 OnLineDAL）；除了这三个层之外，还有一个模型层（取名 OnLineModels）。创建之后打开对应的文件夹，看到里面内容如图 4-4 所示。



图 4-4　在线考试系统项目





4.2　任务实施

4.2.1　用三层架构搭建“在线考试系统”系统框架

1. 创建空白解决方案

(1) 启动 Visual Studio 2005，选择“文件”|“新建”|“项目”命令，打开“新建项目”对话框，如图 4-5 所示。



图 4-5　“新建项目”对话框

(2) 在对话框的“项目类型”选项区域中选择“其他项目类型”节点下的“Visual Studio 解决方案”选项，在“模板”选项区域中选择“空白解决方案”选项，设置“名称”为“OnLine”，设置“位置”为“D:\”，单击“确定”按钮。

创建的空白解决方案如图 4-6 所示。



图 4-6　创建的空白解决方案

2. 搭建用户界面表示层

在空白解决方案中创建一个 ASP. NET 网站 OnLineWeb，作为三层架构中的表示层。

(1) 在解决方案资源管理器窗口右击“解决方案 OnLine”选项，在弹出的快捷菜单中选择“添加”|“新建网站”命令，打开“添加新网站”对话框，如图 4-7 所示。

(2) 在该对话框的“模板”选项区域中选择“ASP. NET 网站”选项，设置“位置”为“文件系统”、“D:\OnLine\OnLineWeb”，设置“语言”为“Visual C＃”，单击“确定”按钮。

搭建的用户界面表示层如图 4-8 所示。

3. 搭建业务逻辑层

(1) 在解决方案资源管理器窗口右击“解决方案 OnLine”选项，在弹出的快捷菜单中选





图 4-7　新建网站

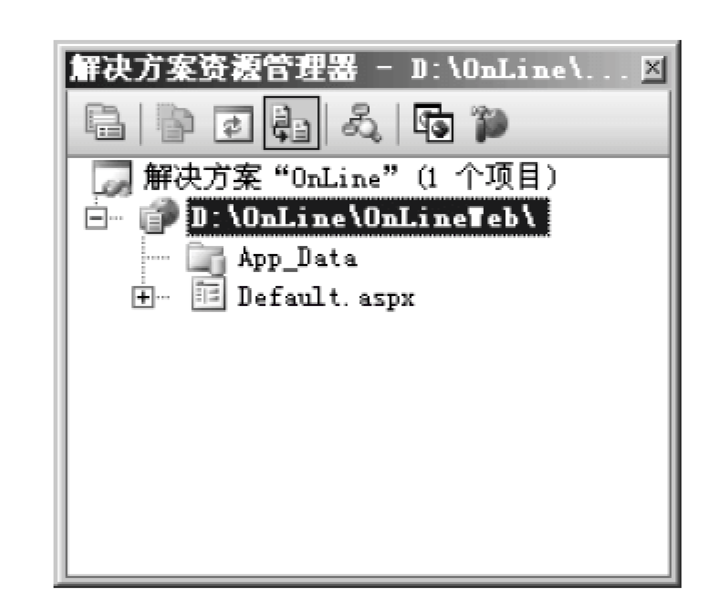


图 4-8　用户界面表示层

择“添加”|“新建项目”命令，打开“添加新项目”对话框。如图 4-9 所示。

(2) 在该对话框的“项目类型”选项区域中选择“Visual C#”选项，在“模板”选项区域中选择“类库”选项，设置名称为“OnLineBLL”，设置目录为“D:\OnLine”，单击“确定”按钮。

搭建的业务逻辑层如图 4-10 所示。



图 4-9　新建业务逻辑层



图 4-10　业务逻辑层

4. 搭建数据访问层

搭建数据访问层的步骤与搭建业务逻辑层类似，数据访问层的名称为 OnLineDAL。

5. 搭建模型层

搭建模型层的步骤也与搭建业务逻辑层类似，模型层的名称为 OnLineModels。

最终搭建的三层架构如图 4-3 所示。

6. 添加各层之间的依赖关系

(1) 右击用户界面表示层 OnLineWeb，在弹出的快捷菜单中选择“添加引用”命令，打开“添加引用”对话框，如图 4-11 所示。



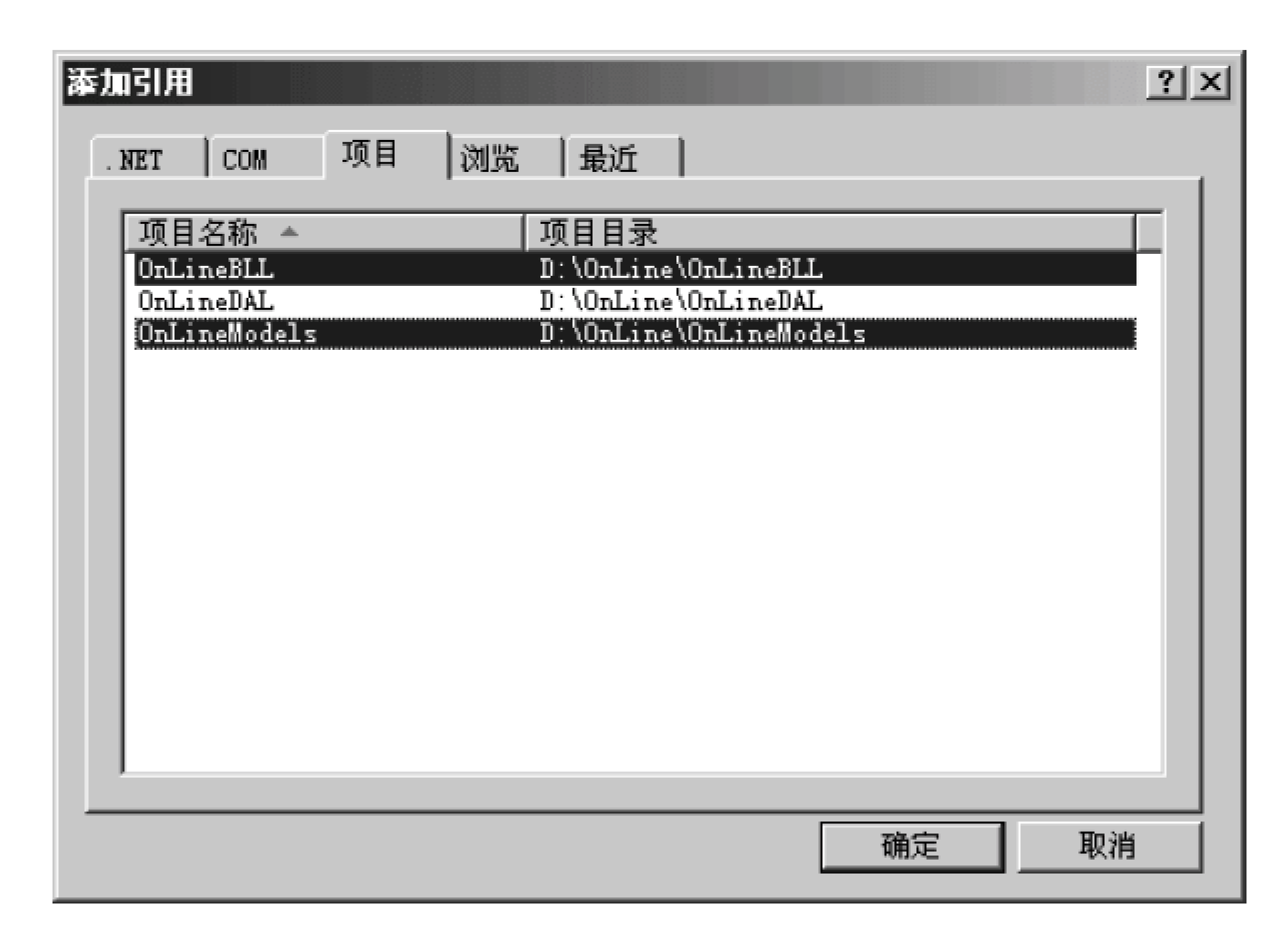


图 4-11　在用户界面表示层添加引用

(2) 在该对话框中选择“项目”选项卡，按下 Ctrl 键选中 OnLineBLL 和 OnLineModels 项目，单击“确定”按钮。

这样用户界面表示层(OnLineWeb)就引用了业务逻辑层(OnLineBLL)和模型层(OnLineModels)。

(3) 用同样的方法在业务逻辑层(OnLineBLL)引用数据访问层(OnLineDAL)和模型层(OnLineModels)；在数据访问层(OnLineDAL)引用模型层(OnLineModels)。

至此，三层架构的“在线考试系统”框架搭建完成。

4.2.2　为在线考试系统创建模型层

1. 在线考试系统模型层介绍

模型层用来存放系统用到的业务实体类。在线考试系统的数据库中有 10 个数据表，对应于模型层中的 10 个实体类，如图 4-12 所示，为了便于识别，实体类的名称和数据表同名。



图 4-12　模型层中的实体类

2. 创建模型层

在模型层中添加业务实体类的步骤如下：

(1) 在解决方案资源管理器窗口右击模型层(OnLineModels)，在弹出的快捷菜单中选择“添加”|“新建项”命令，打开“添加新项”对话框，如图 4-13 所示。

(2) 在“模板”选项区域中选择“类”选项，输入实体类的名称“examAdmin.cs”，单击“添加”按钮，这样就在模型层(OnLineModels)中添加了一个业务实体类 examAdmin。输入如下代码。

```
using System;
using System.Collections.Generic;
```





图 4-13　创建实体类

```
using System.Text;
namespace OnLineModels
{
    #region exam_admin
    ///<summary>
    ///This object represents the properties and methods of a exam_admin.
    ///</summary>
    [Serializable()]
    public class examAdmin
    {
      public examAdmin()
        {
        }

        private int admin_id;
        private string admin_name=String.Empty;
        private string admin_pass=String.Empty;
        private int mgquestions;
        private int mgStudents;
        private int mgSystem;

        #region Public Properties
    ///#region 主要用于编辑器代码的分块,在编译时会被自动删除

        public int Admin_id
        {
            get {return admin_id;}
            set {admin_id=value;}
        }

        public string Admin_name
        {
            get {return admin_name;}
            set {admin_name=value;}
```



```
        }

        public string Admin_pass
        {
            get {return admin_pass;}
            set {admin_pass=value;}
        }

        public int Mgquestions
        {
            get {return mgquestions;}
            set {mgquestions=value;}
        }

        public int MgStudents
        {
            get {return mgStudents;}
            set {mgStudents=value;}
        }

        public int MgSystem
        {
            get {return mgSystem;}
            set {mgSystem=value;}
        }
        #endregion

    }
    #endregion
}
```

用同样的方法添加其他的实体类。

小贴士

#region...endregion是一个分块预处理命令，它主要用于编辑器代码的分块，在编译时会被自动删除。在程序中可以把它们删除掉而不会有什么影响。

练　　习

1. 单项选择题

（1）在三层结构中，用户界面表示层的主要职责是________。

A. 数据处理　　B. 数据展示　　C. 数据传递　　D. 数据存取

（2）在三层结构中，业务逻辑层的主要职责是________。

A. 数据处理和数据存取　　B. 数据处理和数据传递

C. 数据存取和数据展示　　D. 数据展示和数据传递

（3）在三层结构中，数据访问层的主要职责是________。

A. 数据处理　　B. 数据展示　　C. 数据存取　　D. 数据传递



(4) 在三层结构中,用户界面表示层依赖________。
A. 数据访问层　　C. 业务逻辑层
B. 业务逻辑层和数据访问层　　D. 自己

(5) 在三层结构中,业务逻辑层依赖________。
A. 用户界面表示层　　B. 自己
C. 用户界面表示层和业务逻辑层　　D. 数据访问层

(6) 通常使用________返回多个实体对象集合。
A. 数组　　B. string　　C. object　　D. List<T>

(7) 根据数据库中的表定义,实体类由________组成。
A. 字段和方法　　B. 字段和结构
C. 字段和属性　　D. 字段和索引器

(8) 一个实体对象通常封装数据表中________记录。
A. 1 条　　B. 2 条　　C. 3 条　　D. 多条

2. **简答题**

(1) 简述三层结构中各层之间的依赖关系。
(2) 简述在三层结构中使用实体类的优点以及实体类在数据访问层的使用情况。
(3) 画出在三层结构中使用实体类时项目间的依赖关系图。

实　　训

实训目的

熟练掌握三层架构的搭建。

实训内容

(1) 搭建在线考试系统的系统框架。
(2) 在模型层中创建试卷实体类 examPaper. cs。

实训指导

(1) 实训(1)参考课本内容。
(2) 实体类 examPaper. cs 代码如下:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace OnLineModels
{

    #region exam_papers
    ///<summary>
```






```
///This object represents the properties and methods of a exam_paper.
///</summary>
[Serializable()]
public class examPaper
{

    private int question_id;
    private examAllSubject subject;                    //科目
    private examQuestionType question_type;            //试题类型
    private string question_subject=String.Empty;      //试题描述
    private string question_keys=String.Empty;         //试题答案
    private string a=String.Empty;                     //选项
    private string b=String.Empty;
    private string c=String.Empty;
    private string d=String.Empty;
    private string user_key=String.Empty;

    public examPaper()
    {
    }

    #region Public Properties

    public int Question_id
    {
        get {return question_id;}
        set {question_id=value;}
    }

    public examAllSubject examAllSubject
    {
        get {return subject;}
        set {subject=value;}
    }

    public examQuestionType examQuestionType
    {
        get {return question_type;}
        set {question_type=value;}
    }

    public string Question_subject
    {
        get {return question_subject;}
        set {question_subject=value;}
    }

    public string Question_keys
    {
```



```
            get {return question_keys;}
            set {question_keys=value;}
        }

        public string A
        {
            get {return a;}
            set {a=value;}
        }

        public string B
        {
            get {return b;}
            set {b=value;}
        }

        public string C
        {
            get {return c;}
            set {c=value;}
        }

        public string D
        {
            get {return d;}
            set {d=value;}
        }
        public string User_key
        {
            get {return user_key;}
            set {user_key=value;}
        }
        #endregion

    }
    #endregion
}
```

任务5　ADO.NET数据库操作与数据访问层类的创建

技能目标

能使用 ADO.NET 提供的各种数据库操作对象为在线考试系统创建数据访问层的各种类文件。

知识目标

掌握 ADO.NET 提供的各种数据库操作对象的用法，掌握对各类数据库表的增、删、改、查方法。

任务描述

对数据库表的增、删、改、查操作是实现动态网站交互的一种很重要的手段，利用 ADO.NET 提供的各种对象可以实现对数据库的各种操作。

本章任务如下：

- 为在线考试系统创建 DBHelper 类；
- 为在线考试系统创建 StudentService 类。

5.1　知识准备

5.1.1　ADO.NET 简介

ADO.NET 是.NET Framework 中用于数据访问的组件，它由 Microsoft ActiveX Data Objects(ADO)改进而来，是一组用于和数据源进行交互的面向对象类库。通常情况下，数据源是数据库，但它同样也能够是文本文件、Excel 表格或者 XML 文件。ADO.NET 允许和不同类型的数据源以及数据库进行交互。微软公司认为，ADO.NET 是对早期 ADO 技术的“革命性改进”。应该说，它确实是一种非常优秀的数据访问技术，对于使用.NET Framework 进行软件开发的程序员来说，它是必须掌握的技术之一。

ADO.NET 提供与数据源进行交互的相关的公共方法，但是对于不同的数据源采用一组不同的类库。这些类库称为 Data Providers，并且通常是以与之交互的协议和数据源的类型来命名的。



ADO. NET 的对象模型如图 5-1 所示。

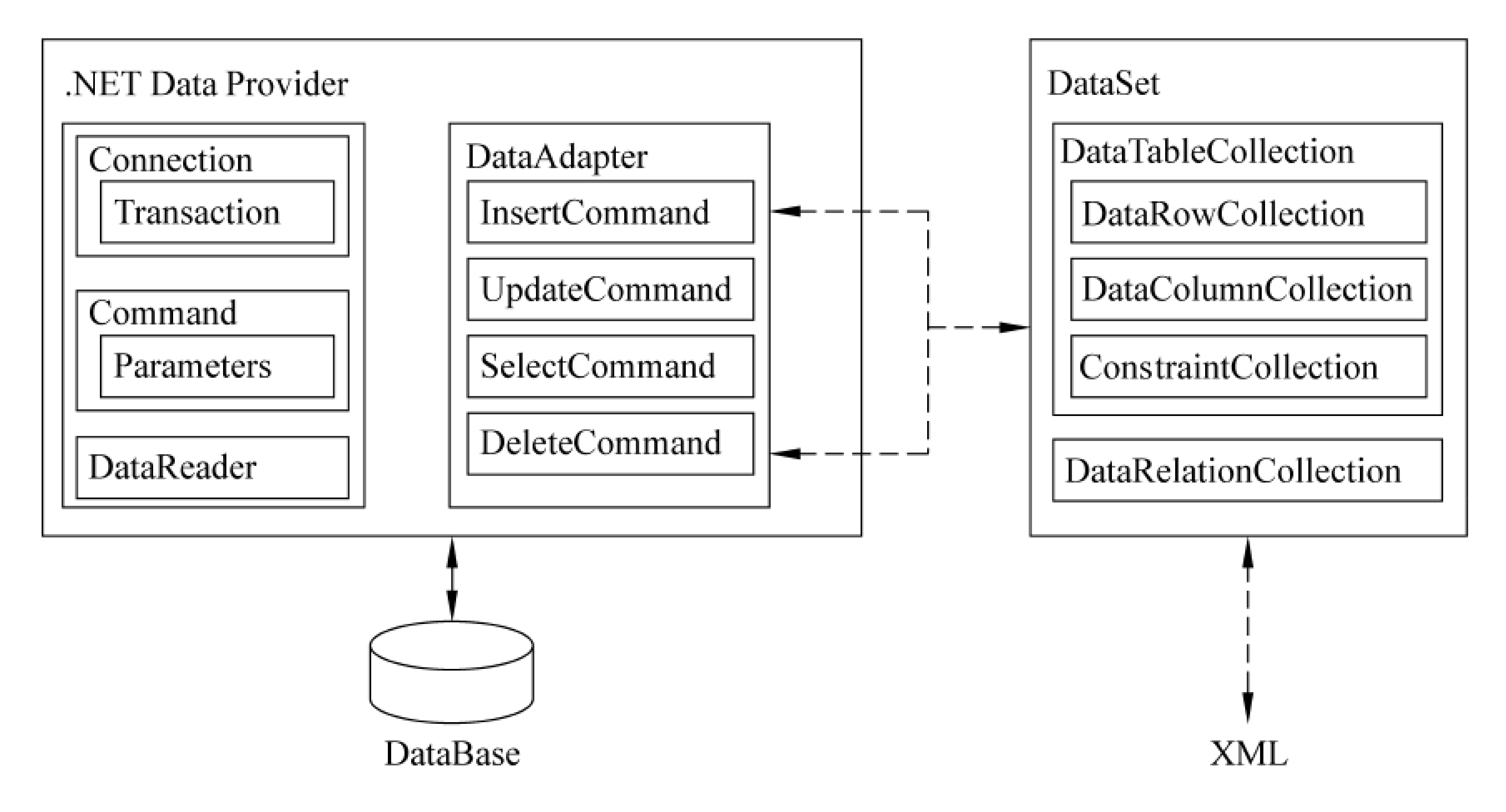


图 5-1　ADO. NET 的对象模型

ADO. NET 包含两个核心组件：DataSet 和. NET Framework。

1. 数据提供程序 .NET Framework

. NET Framework 中的数据提供程序组件用于同数据源打交道。它包含 4 个对象：Connection 对象、Command 对象、DataReader 对象和 DataAdapter 对象。由于数据源不同，上述 4 个对象分别针对不同的数据源做了不同的实现，比如对于 SQL Server 数据库，它们的具体实现是 SqlConnection、SqlCommand、SqlDataReader 和 SqlDataAdapter；对于 Access 数据库，它们的实现是 OleDbConnection、OleDbCommand、OleDbDataReader 和 OleDbDataAdapter。本章以 SQL Server 数据库为主进行介绍。

（1）Connection 对象

Connection 对象主要用于开启程序和数据库之间的联结，它有一个 ConnectionString 属性，用于设置打开数据库的字符串。没有利用联结对象将数据库打开，是无法从数据库中取得数据的，它在 ADO. NET 的最底层。

（2）Command 对象

Command 对象主要用来对数据库发出一些指令，例如可以对数据库下达查询、新增、修改、删除数据等指令，以及呼叫存在数据库中的预存程序等。这个对象架构在 Connection 对象上，它有一个 CommandText 属性，用于设置针对数据源执行的 SQL 语句或存储过程。

（3）DataReader 对象

DataReader 对象用于从数据源获取只进的、只读的数据流。当我们只需要循序地读取数据而不需要其他操作时，可以使用 DataReader 对象。DataReader 对象只是一次一笔向下循序地读取数据源中的数据，而且这些数据是只读的，并不允许做其他的操作。因为 DataReader 在读取数据的时候限制了每次只读取一笔，而且只能只读，所以使用起来不但节省资源而且效率很好。使用 DataReader 对象除了效率较好之外，因为不用把数据全部传回，故可以降低网络的负载。注意它不能用代码直接创建，只能通过 Command 对象的

ExecuteReader 方法来获得。

(4) DataAdapter 对象

DataAdapter 对象架构在 Command 对象上，提供了许多配合 DataSet 使用的功能，是数据提供程序组件中功能最复杂的对象，它是 Connection 对象和数据集之间的桥梁。

2. 数据集 DataSet

DataSet 对象可以视为一个暂存区(Cache)，可以把从数据库中所查询到的数据保存起来，甚至可以将整个数据库显示出来。DataSet 的能力不只是可以储存多个 Table 而已，还可以通过 DataSetCommand 对象取得一些例如主键等的数据表结构，并可以记录数据表间的关联。它有两个集合：DataTableCollection 集合和 DataRelationCollection 集合。其中，DataTableCollection 集合又包含三个集合，分别是行集合 DataRowCollection、列集合 DataColumnCollection 和约束集合 ConstraintCollection。

(1) DataColumnCollection 集合定义了构成数据表的列。

(2) DataRowCollection 集合包含由 DataColumnCollection 集合定义的实际数据。

(3) ConstraintCollection 集合定义了维护数据完整性的约束。

而 DataRelationCollection 集合用于定义表间关系。

.NET Framework 数据是提供程序用于连接数据源、执行 SQL 语句命令和检索数据的基础。检索到的数据既可以直接处理，也可以放入 DataSet 对象中。

5.1.2　Connection 对象

Connection 对象主要用于建立与指定数据源的连接，处理访问数据源时所需要的安全设置。通过 Connection 对象打开数据库连接，是操作数据库的前提和基础。

1. Connection 对象的属性和方法

(1) Connection 对象的常用属性

Connection 对象的常用属性见表 5-1。

表 5-1　Connection 对象的常用属性

属　　性	说　　明
ConnectionString	执行 Open 方法连接数据源的字符串
Database	将要打开数据库的名称
DataSource	包含数据库的位置和文件
State	显示当前 Connection 对象的状态

(2) Connection 对象的常用方法

Connection 对象的常用方法见表 5-2。



表 5-2　Connection 对象的常用方法

方　　法	说　　明
Close	关闭数据库连接
CreateCommand	创建并返回一个与该连接关联的 SqlCommand 对象
Open	打开一个数据库连接
ChangeDatabase	改变当前连接的数据库。需要一个有效的数据库名称

2. 连接字符串

要连接一个 SQL Server 或其他类型的数据库，必须指明连接数据库的种类、数据库服务器的名称、数据库的名称、登录名称和密码等信息，这些信息就构成了连接字符串。

数据库连接字符串常用的参数如下。

- Provider：这个属性用于设置或返回连接提供程序的名称，仅用于 OleDbConnection 对象。
- Connection Timeout：在终止尝试并产生异常前，等待连接到服务器的连接时间长度(以秒为单位)。默认值是 15 秒。
- Initial Catalog 或 Database：数据库的名称。
- Data Source 或 Server：连接打开时使用的 SQL Server 名称，或者是 Microsoft Access 数据库的文件名。
- Password 或 pwd：SQL Server 账户的登录密码。
- User ID 或 uid：SQL Server 登录账户。

只要使用几个主要的参数就可以完成连接数据库的操作。例如：

```
"Server=(local);database=Northwind;uid=sa;pwd=;";
```

这是以 SQL 身份验证登录的写法，如果以 Windows 方式登录则数据库连接字符串应为：

```
Datasource=(local);InitialCatalog=Northwind;Integrated Security=SSPI;
```

连接字符串可以在 Connection 对象的 ConnectionString 属性中指定。

3. Connection 对象的创建

方法 1：利用 SqlConnection 类的无参构造函数创建一个未初始化的 SqlConnection 对象，再用一个连接字符串初始化该对象。程序如下：

```
SqlConnection conn=new SqlConnection();
conn.ConnectionString="Server=(local);database=Northwind;uid=sa;pwd=;";
```

方法 2：利用 SqlConnection 的有参构造函数创建一个 SqlConnection 对象，并为该构造函数的参数指定一个连接字符串。程序如下：

```
SqlConnection conn=new SqlConnection("Data Source=.;Initial Catalog=onLineExam1;User ID=sa;
password=sa")
```



小贴士

在创建 Connection 对象之前，必须先引用 System.Data.SqlClient 和 System.Data 命名空间。

4. 打开和关闭数据库连接

打开和关闭数据库是进行数据库操作必不可少的步骤。打开和关闭数据库是通过连接对象的 Open 和 Close 方法来实现的，也可以由 DataAdapter 对象和 Command 对象自动调用(这两个对象将在后面章节进行介绍)。

下面看两段打开和关闭数据库连接的代码。

连接数据库：

```
SqlConnection conn=new SqlConnection("Data Source=(local);Initial Catalog=Northwind;
Integrated Security=SSPI;");                //创建 SqlConnection 对象,并赋值数据库连接字符串
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        conn.Open();                        //打开与 SqlServer 数据库的连接
        Response.Write("<script>alert('连接成功')</script>");
    }
    catch(SqlException ee)
    {
        Response.Write(ee.Message.ToString());          //抛出异常信息
    }
}
```

断开数据库：

```
protected void Button2_Click(object sender, EventArgs e)
{
try
        {
            conn.Close();                   //断开与 SqlServer 数据库的连接
            Response.Write("<script>alert('已断开连接')</script>");
        }
catch(SqlException ee1)
        {
            Response.Write(ee1.Message.ToString());      //抛出异常信息
        }
}
```

小贴士

OleDbConnection 创建 Connection 对象的方法与 SqlConnectinon 创建 Connection 对象的方法类似，下面来看一段源代码。

```
OleDbConnection con=new OleDbConnection();
con.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=C:\\CSharpSamples\\School.mdb";
con.Open();
```



代码剖析：

第 1 行代码创建 OleDbConnection 对象，该对象是一个通用的连接对象，用它可以打开多种数据源，此处用来创建访问 Access 数据库的连接。

第 2 行和第 3 行是一个语句。代码中的 ConnectionString 属性是 OleDbConnection 对象用于设置打开数据库的字符串，其中，Provider 指定数据提供者，即数据源类型；Data Source 指定 Access 数据库路径及名称。

第 4 行代码调用 OleDbConnection 对象的 Open 方法打开数据库。

5.1.3 Command 对象

没有 ADO.NET 的 Command 对象，数据库不能完成任何工作。不管是要求数据库提供有关数据的信息，还是插入或者更新数据，都要求从一个 Command 对象开始。Command 对象提供对数据源执行 SQL 命令的接口，可以用来对数据库发出一些指令。

1. Command 对象的属性和方法

下面介绍 Command 对象的常用属性和方法。

（1）Command 对象的常用属性

Command 对象的常用属性见表 5-3。

表 5-3 Command 对象的常用属性

属　　性	说　　明
CommandText	获取或设置对数据源执行的 SQL 语句或者存储过程名或表名
Connection	获取或设置此 Command 对象使用的 Connection 对象的名称

（2）Command 对象的常用方法

Command 对象的常用方法见表 5-4。

表 5-4 Command 对象的常用方法

方　　法	说　　明
ExecuteNonQuery	执行 SQL 语句并返回受影响的行数。ExecuteNonQuery 方法执行更新操作，诸如那些与 UPDATE、INSERT 和 DELETE 语句有关的操作，在这些情况下，返回值是命令影响的行数。对于其他类型的语句，诸如 SET 或 CREATE 语句，则返回值为－1
ExecuteScalar	执行查询，并返回查询所返回的结果集中第一行的第一列，忽略其他列或行。如果只想检索数据库信息中的一个值，而不需要返回表或数据流形式的数据库信息，例如，只需要返回 COUNT(＊)、SUM(Price) 或 AVG(Quantity) 等聚合函数的结果，那么 Command 对象的 ExecuteScalar 方法就很有用
ExecuteReader	执行 SELECT 语句并返回数据集。ExecuteReader 方法通常与查询命令一起使用，并且返回一个数据阅读器对象 SqlDataReader 类的一个实例。如果通过 ExecuteReader 方法执行一个更新语句，则该命令成功地执行，但是不会返回任何受影响的数据行
ChangeDatabase	改变当前连接的数据库。需要一个有效的数据库名称

2. Command对象的创建

建立数据连接以后，可以利用 Command 对象来执行命令并从数据源返回结果。Command 对象的创建是由构造函数完成的，Command 对象常用的构造函数包括两个参数，一个是要执行的 SQL 语句，另一个是已经建立的 Connection 对象，程序如下：

```
SqlConnection connection=new SqlConnection("Data Source=.;Initial Catalog=onLineExam1;User ID=sa;
password=sa");
connection.Open();
String safeSql="select * from student";
SqlCommand cmd=new SqlCommand(safeSql, conn);
```

3. 示例

将 Student 表中学号为 090001 学生的年龄改为 18 岁。

源代码如下：

```
1  SqlConnection con=new SqlConnection();
2  con.ConnectionString="Data Source=(local);Initial Catalog=onLineExam1;
   uid=sa;pwd=sa";
3  con.Open();
4  int age=18;
5  string sno="090001";
6  string sql="Update Student Set 年龄="+age+" Where 学号='"+sno+"'";
7  SqlCommand cmd=new SqlCommand(sql,con);
8  cmd.ExecuteNonQuery();
```

代码剖析：

第 6 行建立一个动态的 SQL 语句，这是编程时常用的一种手法。

第 7 行代码创建 SqlCommand 对象，其中，sql 参数指定针对数据源要执行的 SQL 语句，con 参数为指向已打开数据源的连接对象。

第 8 行代码调用 SqlCommand 对象的 ExecuteNonQuery 方法，该方法通常用于执行 UPDATE、INSERT 和 DELETE 语句，返回值为该命令所影响的行数。

小贴士

根据所用的.NET Framework 数据提供程序的不同，Command 对象也可以分成 4 种，分别是 SqlCommand、OleDbCommand、OdbcCommand 和 OracleCommand，在实际的编程过程中应根据访问的数据源不同，选择相应的 Command 对象。

5.1.4　DataReader对象

当 Command 对象返回结果集时，需要使用 DataReader 对象来检索数据。DataReader 对象返回一个来自 Command 的只读的、只能向前的数据流。



1. DataReader 对象常用属性和方法

(1) DataReader 对象的常用属性

DataReader 对象的常用属性见表 5-5。

表 5-5　DataReader 对象的常用属性

属　性	说　　明
FieldCount	获取当前行的列数
Item	索引器属性，以原始格式获得一列的值

(2) DataReader 对象的常用方法

DataReader 对象的常用方法见表 5-6。

表 5-6　DataReader 对象的常用方法

方　法	说　　明
Read	使 DataReader 对象前进到下一条记录(如果有)，使用 DataReader 对象中的 Read 方法来遍历整个结果集，不需要显式地向前移动指针，或者检查文件的结束。如果没有要读取的记录了，则 Read 方法会自动返回 false
Close	关闭 DataReader 对象
Get	用来读取数据集的当前行的某一列的数据

2. DataReader 对象的创建

前面讲过，Connection 对象和 Command 对象都是利用构造函数来创建的，但是没有构造函数创建 DataReader 对象。通常使用 Command 类的 ExecuteReader 方法来创建 DataReader 对象，方法如下：

```
SqlCommand cmd=new SqlCommand(commandText,ConnectionObject)
SqlDataReader dr=cmd.ExecuteReader();
```

常常用 While 循环来遍历 DataReader 中的记录，方法如下：

```
While(dr.Read())
{
    //do something with the current record
}
```

ExecuteReader 方法返回 DataReader 对象时，当前光标的位置是在第一条记录的前面。必须调用数据阅读器的 Read()方法把光标移动到第一条记录，然后第一条记录就是当前记录。如果阅读器包含的记录不止一条，Read()方法返回一个 bool 值 true。也就是说 Read()方法的作用是在允许范围内移动光标位置到下一条记录，如果当前光标指示着最后一条记录，此时调用 Read()方法得到 false 。

3. 示例

下面的代码将循环显示所有学生的学号和姓名。





源代码如下：

```
1  SqlConnection con=new SqlConnection();
2  con.ConnectionString="Data Source=(local);Initial Catalog=onLineExam1;
   uid=sa;pwd=sa";
3  con.Open();
4  string sql="Select 学号,姓名 from Student";
5  SqlCommand cmd=new SqlCommand(sql,con);
6  SqlDataReader dr=cmd.ExecuteReader();
7  while(dr.Read())
8  {
9  MessageBox.Show(dr["学号"].ToString()+" "+dr["姓名"].ToString());
10 }
```

代码剖析：

第 4 行代码建立 SQL 语句，用于查询 Student 表中所有学生的学号和姓名。

第 5 行代码创建 SqlCommand 对象。

第 6 行代码创建 SqlDataReader 对象，该对象可以存储数据源数据。

第 7 行代码调用 SqlDataReader 对象的 Read()方法，该方法使 SqlDataReader 前进到下一条记录。

小贴士

每次使用完 DataReader 对象后，都应调用 Close()方法。

5.1.5 DataSet 对象

DataSet 是一种容器，可以由从数据适配器执行的 SQL 命令或存储过程所填充。它不直接绑定到数据源，可以缓存来自多个数据源的数据。DataSet 的设计是为了实现独立于任何数据源(包含数据库、XML 数据源)的数据访问。DataSet 的主要特性如下：

(1) 独立性。DataSet 独立于各种数据源。微软公司在推出 DataSet 时就考虑到各种数据源的多样性、复杂性。在.NET 中，无论什么类型的数据源，都会提供一致的关系编程模型，而这就是 DataSet。

(2) 离线(断开)和连接 DataSet 既可以以离线方式，也可以以实时连接来操作数据库中的数据。这一点有点像 ADO 中的 RecordSet。DataSet 对象是一个可以用 XML 形式表示的数据视图，是一种数据关系视图。

DataSet 对象的创建可以由构造函数来完成，创建 DataSet 对象的方法如下：

```
DataSet mydataset=new DataSet();
```

DataSet 是数据库中的数据在本地计算机中映射成的缓存，它是一个完整的数据集，由大量相关的数据结构组成。在 DataSet 内部，主要可以存储 5 种对象。

- DataTable：使用行、列形式来组织的一个表格式数据集。
- DataColumn：一个规则的集合，描述决定将什么数据存储到一个 DataRow 中。
- DataRow：由单行数据库数据构成的一个数据集合，该对象是实际的数据存储。
- Constraint：决定能进入 DataTable 的数据。



✍ DataRelation：描述了不同的 DataTable 之间如何关联。

在 DataSet 内部是一个或多个 DataTable 的集合。

5.1.6 DataAdapter 对象

DataAdapter(数据适配器)对象是一种用来充当 DataSet 对象与实际数据源之间桥梁的对象。DataSet 对象是一个非连接的对象，它与数据源无关；而 DataAdapter 则正好负责填充它并把它的数据提交给一个特定的数据源，它与 DataSet 配合使用，可以执行新增、查询、修改和删除等多种操作。

DataAdapter 对象是一个双向通道，用来把数据从数据源中读到一个内存表中，以及把内存中的数据写回到一个数据源中。这两种操作分别称作填充(Fill)和更新(Update)。

DataAdapter 对象可以建立并初始化数据表(即 DataTable)，对数据源执行 SQL 指令，与 DataSet 对象结合，提供 DataSet 对象存取数据。其主要的工作流程是由 Connection 对象建立与数据源联机，DataAdapter 对象经由 Command 对象操作 SQL 指令以存取数据，存取的数据通过 Connection 对象返回给 DataAdapter 对象，DataAdapter 对象将数据放入其所产生的 DataTable 对象，将 DataAdapter 对象中的 DataTable 对象加入到 DataSet 对象中的 DataTables 对象中。

1. DataAdapter 对象的属性、方法和事件

(1) DataAdapter 对象的常用属性

DataAdapter 对象的常用属性见表 5-7。

表 5-7　DataAdapter 对象的常用属性

属　性	说　明
DeleteCommand	获取或设置一个 Transact-SQL 语句或存储过程，以从数据集删除记录
InsertCommand	获取或设置一个 Transact-SQL 语句或存储过程，以在数据源中插入新记录
SelectCommand	获取或设置一个 Transact-SQL 语句或存储过程，用于在数据源中选择记录
UpdateCommand	获取或设置一个 Transact-SQL 语句或存储过程，用于更新数据源中的记录
TableMappings	SqlDataAdapter 用来将查询的结果映射到 DataSet 的信息集合

(2) DataAdapter 对象的常用方法

DataAdapter 对象的常用方法见表 5-8。

表 5-8　DataAdapter 对象的常用方法

方　法	说　明
Fill	执行存储于 SelectCommand 中的查询，并将结果存储在 DataTable 中
Update	向数据库提交存储在 DataSet(或 DataTable、DataRows)中的更改。该方法会返回一个整数值，其中包含着在数据存储中成功更新的行数
SelectCommand	获取或设置一个 Transact-SQL 语句或存储过程，用于在数据源中选择记录

（3）DataAdapter 对象的常用事件

DataAdapter 对象的常用事件见表 5-9。

表 5-9　DataAdapter 对象的常用事件

方　　法	说　　明
FillError	当 DataAdapter 遇到填充 DataSet 或 DataTable 的一个错误时，该事件被触发
RowUpdated	向数据库提交一个修改的行之后被触发
RowUpdating	向数据库提交一个修改的行之前被触发

2．DataAdapter 对象的创建

DataAdapter 对象表示用于填充 DataSet 和更新数据库的一组数据命令和一个数据库连接。

DataAdapter 对象的创建是由构造函数完成的。DataAdapter 对象常用的构造函数包括两个参数，一个是要执行的 SQL 语句，另一个是已经建立的 Connection 对象，程序如下：

```
SqlDataAdapter da=new SqlDataAdapter("Select * From Student",strConn);
DataSet ds=new DataSet();
da.Fill(ds);                              //这里 ds 中的表名为 Table
```

3．示例

（1）从数据源获得数据

SqlDataAdapter 对象有一个 SelectCommand 属性，它封装一个 SqlCommand，通过调用 SqlDataAdapter 对象的 Fill 方法，可以将数据源的数据传输到客户端，并存储到数据集中。

源代码如下：

```
1  SqlConnection con=new SqlConnection();
2  con.ConnectionString="Data Source=(local);Initial
                        Catalog=onLineExam1;uid=sa;pwd=sa";
3  con.Open();
4  SqlDataAdapter da=new SqlDataAdapter();
5  SqlCommand cmd=new SqlCommand("Select * From Student",con);
6  da.SelectCommand=cmd;
7  DataSet ds=new DataSet();
8  da.Fill(ds,"stu");
```

代码剖析：

第 4 行代码创建 SqlDataAdapter 对象。

第 5 行代码创建 SqlCommand 对象，该对象指定了 SQL 语句和活动连接。

第 6 行代码很重要，它设置了 SqlDataAdapter 对象的 SelectCommand 属性。

第 7 行代码创建 DataSet 对象。



第 8 行代码调用 SqlDataAdapter 对象的 Fill 方法从数据源读取数据并将其填充到数据集中，其中 Stu 是数据集中的表的名称。

（2）更新数据源数据

所谓“更新数据源数据”，包括将客户端数据插入到数据源、修改数据源的现存数据和删除数据源的现存数据。

添加数据：将一位学生的信息插入到 Student 表中。

源代码如下：

```
1   SqlConnection con=new SqlConnection();
2   con.ConnectionString="Data Source=(local);Initial
                          Catalog=onLineExam1;uid=sa;pwd=sa";
3   con.Open();
4   SqlDataAdapter da=new SqlDataAdapter("Select * From Student",con);
5   SqlCommandBuilder builder=new SqlCommandBuilder(da);
6   DataSet ds=new DataSet();
7   da.Fill(ds,"Stu");
8   DataRow dr=ds.Tables["Stu"].NewRow();
9   dr["学号"]="090011";
10  dr["姓名"]="刘晓念";
11  dr["性别"]="男";
12  dr["年龄"]=19;
13  dr["系别"]="计科系";
14  ds.Tables["Stu"].Rows.Add(dr);
15  da.Update(ds,"Stu");
```

代码剖析：

第 4 行代码创建 SqlDataAdapter 对象。

第 5 行代码很重要，它创建了一个 SqlCommandBuilder 对象，它实例化时使用 SqlDataAdapter 对象作为参数。如果设置了 SqlDataAdapter 对象的 SelectCommand 属性，则可以创建一个 SqlCommandBuilder 对象来自动生成用于单表更新的 Transact-SQL 语句。为了生成 INSERT、UPDATE 或 DELETE 语句，SqlCommandBuilder 会自动使用 SelectCommand 属性来检索所需的元数据集。

第 6 行代码创建数据集。

第 7 行代码调用 SqlDataAdapter 对象的 Fill 方法填充数据集。

第 8 行代码创建数据集中 Stu 表的新行。

第 9～13 行代码创建新行的各列值。

第 14 行代码将新行添加到数据集的 Stu 表中。

第 15 行代码调用 SqlDataAdapter 对象的 Update 方法更新数据源，即将新行物理地添加到数据库中。

修改数据：将学号为 090011 的学生的年龄改为 20 岁。

源代码如下：

```
SqlConnection con=new SqlConnection();
con.ConnectionString="Data Source=myserver;Initial
```



```
                        Catalog=School;uid=abc;pwd=abc";
3   con.Open();
4   SqlDataAdapter da=new SqlDataAdapter("Select * From Student",con);
5   SqlCommandBuilder builder=new SqlCommandBuilder(da);
6   DataSet ds=new DataSet();
7   da.Fill(ds,"Stu");
8   DataRow[] drs=ds.Tables["Stu"].Select("学号='090011'");
9   drs[0]["年龄"]=20;
10  da.Update(ds,"stu");
```

代码剖析：

第 8 行代码调用 DataTable 对象的 Select 方法，它返回与筛选条件相匹配的所有 DataRow 数组。

第 9 行代码中的 drs[0]表示 DataRow 对象数组中的第一行，因为学号是主键，所以 Select 方法至多返回一行，drs[0]就表示学号为 090011 的那一行记录。

删除数据：删除 Student 表中学号为 090011 的学生记录。

源代码如下：

```
1   SqlConnection con=new SqlConnection();
2   con.ConnectionString="Data Source=myserver;Initial
                        Catalog=School;uid=abc;pwd=abc";
3   con.Open();
4   SqlDataAdapter da=new SqlDataAdapter("Select * From Student",con);
5   SqlCommandBuilder builder=new SqlCommandBuilder(da);
6   DataSet ds=new DataSet();
7   da.Fill(ds,"Stu");
8   DataRow[] drs=ds.Tables["Stu"].Select("学号='090011'");
9   drs[0].Delete();
10  da.Update(ds,"stu");
```

代码剖析：

第 9 行代码中的 drs[0]表示学号为 090011 的那一行记录，DataRow 对象的 Delete 方法将该行从 Stu 表中删除。

第 10 行代码调用 SqlDataAdapter 对象的 Update 方法更新数据源，即将数据库中学号为 090011 的那行记录物理地删除。

5.2　任务实施

5.2.1　为在线考试系统数据访问层创建 DBHelper 类

在对数据库的所有操作中，数据库的打开、连接、执行 SQL 语句是很常见的操作，而且这些操作经常要反复执行。如果每次都编写代码，势必会浪费很多精力，因此我们开发出一个执行数据库操作的数据库操作类 DBHelper 类用于执行这些重复的操作。



1. 创建数据访问层

（1）在解决方案资源管理器窗口右击数据访问层（OnLineDAL），在弹出的快捷菜单中选择“添加”|“新建项”命令，打开“添加新项”对话框，如图 5-2 所示。



图 5-2　“添加新项”对话框

（2）在该对话框中选择“类”选项，输入类的名称“ConnDBHelper . cs”，单击“添加”按钮。

创建之后代码如下：

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  namespace OnLineDAL
5      {
6      class ConnDBHelper
7      {
8      }
9      }
```

现在 ConnDBHelper 类中还没有内容。

2. 添加引用命名空间

为了使用 SQL Server 的常规操作，需要在该类中添加对 System. Data. SqlClient 和 System. Data 命名空间的引用。

在第 3 行后输入如下内容。

```
using System.Data;
using System.Data.SqlClient;
```

3. 添加数据库连接方法

在 ConnDBHelper 类中添加连接数据库的方法。



```
public  class ConnDBHelper
{
  private static SqlConnection connection;
  public static SqlConnection Connection
  {
   get
   {
     string connectionString=
               "Data Source=.;InitialCatalog=onlineExam1;UserID=sa;password=sa";
       if (connection==null)    // 根据连接字符串的状态，打开或关闭连接
       {
         connection=new SqlConnection(connectionString);
         connection.Open();
       }
      else if (connection.State==System.Data.ConnectionState.Closed)
          {
            connection.Open();
          }
          else if (connection.State==System.Data.ConnectionState.Broken)
               {
                   connection.Close();
                   connection.Open();
               }
               return connection;
   }
  }
}
```

4. 添加其他方法和函数

在 Connection 方法后添加其他方法和函数。

```
public  class ConnDBHelper
{
  ...
    public static int ExecuteCommand(string safeSql)   // 执行SQL语句，并使用ExecuteNonQuery返回受影响行数
        {
            SqlCommand cmd=new SqlCommand(safeSql, Connection);
            int result=cmd.ExecuteNonQuery();
            return result;
        }
        public static int ExecuteCommand(string sql, params SqlParameter[] values)   // 上面方法的重载
        {
            SqlCommand cmd=new SqlCommand(sql, Connection);
            cmd.Parameters.AddRange(values);
            return cmd.ExecuteNonQuery();
        }
        public static int GetScalar(string safeSql)   // 执行SQL语句，并使用ExecuteScalar返回第一行第一列的值
        {
            SqlCommand cmd=new SqlCommand(safeSql, Connection);
            int result=Convert.ToInt32(cmd.ExecuteScalar());
```



```
            return result;
        }
    public static int GetScalar(string sql, params SqlParameter[] values)
    {
        SqlCommand cmd=new SqlCommand(sql, Connection);
        cmd.Parameters.AddRange(values);
        int result=Convert.ToInt32(cmd.ExecuteScalar());
        return result;
    }
    public static SqlDataReader GetReader(string safeSql)
    {
        SqlCommand cmd=new SqlCommand(safeSql, Connection);
        SqlDataReader reader=cmd.ExecuteReader();
        return reader;
    }
    public static SqlDataReader GetReader(string sql, params SqlParameter[]
    values)
    {
        SqlCommand cmd=new SqlCommand(sql, Connection);
        cmd.Parameters.AddRange(values);
        SqlDataReader reader=cmd.ExecuteReader();
        return reader;
    }
    public static DataTable GetDataSet(string safeSql)
    {
        DataSet ds=new DataSet();
        SqlCommand cmd=new SqlCommand(safeSql, Connection);
        SqlDataAdapter da=new SqlDataAdapter(cmd);
        da.Fill(ds);
        return ds.Tables[0];
    }
    public static DataTable GetDataSet(string sql, params SqlParameter[] values)
    {
        DataSet ds=new DataSet();
        SqlCommand cmd=new SqlCommand(sql, Connection);
        cmd.Parameters.AddRange(values);
        SqlDataAdapter da=new SqlDataAdapter(cmd);
        da.Fill(ds);
        return ds.Tables[0];
    }
}
```

上面方法的两个函数的重载

执行SQL语句，并使用ExecuteReader返回reader对象

上面方法的两个函数的重载

执行SQL语句，并返回一个DataSet对象

上面方法的两个函数的重载

这个类中包括了最常用的读取数据库的方法，当需要执行相应的操作的时候，只需实例化一个对象，并调用其中的方法即可。

5.2.2 创建StudentService.cs类

数据访问层包括了各个数据库表的增、删、改、查方法，针对每一个数据表，对应着相应的增、删、改、查方法。



考生信息表是数据库中一种重要的数据表，因此，需要创建一个类，实现对该表的增、删、改、查操作。

创建 StudentService.cs 类的方法与创建 DBHelper 类相同，内容如下：

```
public static partial class StudentService
{
  //添加信息
  public static void AddStudent(examStudent student)
  {
    string sql="INSERT exam_students (stu_id, stu_name, class_id, isLogin, isSubmit,stu_pwd)
    VALUES (@stuId, @stuName, @classId, @isLogin, @isSubmit,@stuPwd)";

    try
    {
      SqlParameter[] para=new SqlParameter[]
      {
        new SqlParameter("@stuId", student.Stu_id),
        new SqlParameter("@stuName", student.Stu_name),
        new SqlParameter("@stuPwd", student.Stu_pwd),
        new SqlParameter("@classId", student.examStuClass.Class_id),    //外键
        new SqlParameter("@isLogin", student.IsLogin),
        new SqlParameter("@isSubmit", student.IsSubmit)
      };
       ConnDBHelper.ExecuteCommand(sql, para);
    }
       catch (Exception e)
       {
          Console.WriteLine(e.Message);
          throw e;
       }
  }
        //获取所有信息
        public static IList<examStudent> GetAllStudent()
        {
            string sql="select * from exam_students";
            return GetStudentBySql(sql);
        }
        //根据 ID 获取信息
        public static examStudent GetStudentById(string  sId)
        {
            string sql="select * from exam_students WHERE stu_id=@StuId";
            int classId;
            try
            {
                SqlDataReader reader=ConnDBHelper.GetReader(sql, new
                                         SqlParameter("@StuId", sId));
                if (reader.Read())
                {
                    examStudent student=new examStudent();
                    student.Stu_id=(string)reader["stu_id"];
```



```
                student.Stu_name= (string)reader["stu_name"];
                student.Stu_pwd= (string)reader["stu_pwd"];
                student.IsLogin= (int)reader["isLogin"];
                student.IsSubmit= (int)reader["isSubmit"];
                classId= (int)reader["class_id"];           //外键
                reader.Close();                              //关闭 reader
                student.examStuClass=StuClassService.GetStuClassById(classId);
                                                             //外键处理
                return student;
            }
            else
            {
                reader.Close();
                return null;
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            throw e;
        }
    }
    //根据 SQL 语句获取信息
    //不带参数的 SQL
    private static IList<examStudent> GetStudentBySql(string sql)
    {
        List<examStudent> list=new List<examStudent>();
        try
        {
            DataTable table=ConnDBHelper.GetDataSet(sql);
            foreach (DataRow row in table.Rows)
            {
                examStudent student=new examStudent();
                student.Stu_id= (string)row["stu_id"];
                student.Stu_name= (string)row["stu_name"];
                student.Stu_pwd= (string)row["stu_pwd"];
                student.IsLogin= (int)row["isLogin"];
                student.IsSubmit= (int)row["isSubmit"];
                student.examStuClass=StuClassService.GetStuClassById((int)
                row["class_id"]);                                         //外键处理
                list.Add(student);
            }
            return list;
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            throw e;
        }
    }
    //根据 SQL 语句获取信息
```

```
//带参数的 SQL
private static IList<examStudent>GetStudentBySql(string sql, params
SqlParameter[] values)
{
    List<examStudent>list=new List<examStudent>();
    try
    {
        DataTable table=ConnDBHelper.GetDataSet(sql, values);
        foreach (DataRow row in table.Rows)
        {
            examStudent student=new examStudent();
            student.Stu_id=(string)row["stu_id"];
            student.Stu_name=(string)row["stu_name"];
            student.Stu_pwd=(string)row["stu_pwd"];
            student.IsLogin=(int)row["isLogin"];
            student.IsSubmit=(int)row["isSubmit"];
            student.examStuClass=StuClassService.GetStuClassById((int)
            row["class_id"]);                                    //外键处理
            list.Add(student);
        }
        return list;
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw e;
    }
}
//删除信息
public static void DeleteExamStudentById(string sId)
{
    string sql="DELETE exam_students WHERE stu_id=@Id";
    try
    {
        SqlParameter[] para=new SqlParameter[]
        {
            new SqlParameter("@Id", sId)
        };
        ConnDBHelper.ExecuteCommand(sql, para);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw e;
    }
}
//修改信息
public static void ModifyExamStudent(examStudent student)
{
    string sql=
        "UPDATE exam_students  SET stu_name=@stuName, "+
```



```
                    "class_id=@classId, "+
                    "stu_pwd=@stuPwd, "+
                    "isLogin=@isLogin, "+
                    "isSubmit=@isSubmit"+
                   "WHERE stu_id=@stuId";
            try
            {
                SqlParameter[] para=new SqlParameter[]
                {
                    new SqlParameter("@stuId", student.Stu_id),
                    new SqlParameter("@stuName", student.Stu_name),
                    new SqlParameter("@stuPwd", student.Stu_pwd),
                    new SqlParameter("@classId", student.examStuClass.Class_id),
                    new SqlParameter("@isLogin", student.IsLogin),
                    new SqlParameter("@isSubmit", student.IsSubmit)

                };
                ConnDBHelper.ExecuteCommand(sql, para);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                throw e;
            }
        }
    }
```

该类中所包含的方法如下。

(1) 增的方法

public static void AddStudent(examStudent student)：添加新用户，返回添加后的用户对象。

(2) 查的方法

public static IList<examStudent> GetAllStudent()：得到所有考生信息，返回 IList，用来存放考生信息。

public static examStudent GetStudentById(string sId)：按照考生学号得到考生信息。

private static IList<examStudent> GetStudentBySql(string sql)：提供用于执行的 SQL 语句并返回考生信息对象集合的方法。

private static IList<examStudent> GetStudentBySql(string sql，params SqlParameter[] values)：此方法是上述方法的重载，增加了参数 values。

(3) 删的方法

public static void DeleteExamStudentById(string sId)：按照考生 ID 删除考生信息。

(4) 改的方法

public static void ModifyExamStudent(examStudent student)：更改考生信息，以考生对象为参数。

小贴士

数据访问层中各个类一般命名为“数据库表名+Service”。

除了StudentService类以外，还有其他的类用于实现对不同数据库表的增、删、改、查操作，这些类包括以下几种。

- AdminService.cs：用于操作管理员信息。
- AllQuestionsService.cs：用于操作所有的问题。
- AllSubjectService.cs：用于操作所有的科目。
- ExamPaperService.cs：用于操作考试卷。
- PapermgService.cs：用于操作试卷信息。
- QuestionTypeService.cs：用于操作问题类型。
- StuClassService.cs：用于操作学生班级。
- StuScoreService.cs：用于操作学生成绩。

这些类中都包含了相应数据表的增、删、改、查方法。

练　　习

1. 单项选择题

(1) ADO .NET 的核心对象不包括________。

A. Response　　B. Connection

C. Command　　D. DataReader

(2) Connection对象的________属性用于获取或设置用于打开数据库的字符串。

A. ConnectionString　　B. ConnectionTimeout

C. DataSource　　D. Provider

(3) 下面SqlCommand对象方法中，可以连接SQL语句并返回受影响的行数的方法是________。

A. ExecuteNonQuery　　B. ExecuteScalar

C. ExecuteReader　　D. Connection

(4) ________对象没有自己的构造函数。

A. DataAdapter　　B. Connection

C. Command　　D. DataReader

(5) DataAdapter对象用来填充DataSet的方法是________。

A. Open　　B. Get　　C. Close　　D. Fill

2. 简答题

(1) ADO.NET有哪些对象？其作用各是什么？

(2) 简述访问数据库的总体流程。



实　　训

实训目的

（1）熟练掌握 ADO.NET 对象的使用方法。

（2）会应用各种 ADO.NET 对象创建数据访问层中的类。

实训内容

（1）在 OnLineDAL 类库上右击，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，在里面选择类，并命名为 ExamPaperService.cs。

（2）仿照 StudentService 类在该类中添加增、删、改、查方法。

实训指导

ExamPaperService.cs 类文件主要是操作考试卷，实现数据表的增、删、改、查，创建的时候可以仿照 StudentService 类中的相应方法，调用 ADO.NET 对象来实现，参考代码如下：

```
public static partial class ExamPaperService
{
    //添加新试题
    public static examPaper AddQuestion(examPaper examQuestion,string paperName)
    {
    string sql="insert"+paperName+"(subject_id, question_type,question_subject, question_
              keys, a, b, c, d, user_key)"+"VALUES (@subjectId, @questionType, @
              questionSubject, @questionKeys, @A, @B, @C, @D,@User_key)";
    sql+="; SELECT @@IDENTITY";
    SqlParameter[] para=new SqlParameter[]
    {
      new SqlParameter("@subjectId", examQuestion.examAllSubject.Subjectt_id),      //外键
      new SqlParameter("@questionType", examQuestion.examQuestionType.Type_id),
                                                                                    //外键
      new SqlParameter("@questionSubject", examQuestion.Question_subject),
      new SqlParameter("@questionKeys", examQuestion.Question_keys),
      new SqlParameter("@A", examQuestion.A),
      new SqlParameter("@B", examQuestion.B),
      new SqlParameter("@C", examQuestion.C),
      new SqlParameter("@D", examQuestion.D),
      new SqlParameter("@User_key",examQuestion.User_key)
     };
      int newQuestionId=ConnDBHelper.GetScalar(sql, para);
      return GetExamQuestionById(newQuestionId);
     }
    //根据 id 查询单个试题
    public static examPaper GetExamQuestionById(int questionId)
    {
      string sql="select * from exam_papers WHERE question_id=@questionId";
```



```
        int subjectId;
        int questionType;
        //使用 using 指令及时释放资源
        using(SqlDataReader reader=
              ConnDBHelper.GetReader(sql, new SqlParameter("@questionId", questionId)))
        {
          if (reader.Read())
          {
            examPaper examQuestion=new examPaper();
            examQuestion.Question_id=(int)reader["question_id"];
            examQuestion.Question_subject=(string)reader["question_subject"];
            examQuestion.Question_keys=(string)reader["question_keys"];
            examQuestion.A=Convert.ToString(reader["a"]);
            examQuestion.B=Convert.ToString(reader["b"]);
            examQuestion.C=Convert.ToString(reader["c"]);
            examQuestion.D=Convert.ToString(reader["d"]);
            examQuestion.User_key=Convert.ToString(reader["user_key"]);
            subjectId=(int)reader["subject_id"];              //外键
            questionType=(int)reader["question_type"];        //外键
            //及时关闭 reader
            reader.Close();
            examQuestion.examAllSubject=AllSubjectService.GetSubjectById
            (subjectId);
            examQuestion.examQuestionType=QuestionTypeService.GetQuestionTypeById
            (questionType);
            return examQuestion;
          }
            else
            {
              reader.Close();
              return null;
            }
          }
        }
        //创建试卷表
        public static void CreatePaper(string strCreateTable)
        {
          int result=ConnDBHelper.ExecuteCommand(strCreateTable);
        }
}
```

任务6　验证控件与用户登录

技能目标

能使用各种验证控件为在线考试系统实现各种验证效果。

知识目标

掌握HTML控件的用法，掌握标准服务器控件的用法，掌握验证控件的用法；会使用各种验证控件实现不同的验证效果。

任务描述

在需要用户输入的界面中，有些信息不能随意输入，为了防止用户输入错误，需要提示用户的输入是否正确，控制错误的输入。ASP.NET提供的验证控件可以帮助实现输入的验证功能。

本章任务如下：

◆ 使用各种HTML控件和标准服务器控件布局用户界面；

◆ 使用各种验证控件实现用户输入的验证。

6.1　知识准备

6.1.1　ASP.NET控件基础

控件是ASP.NET中用来布局页面的组件，在Visual Studio 2005中按照控件的类型将其分成9个类，如图6-1所示。除了标准的HTML元素外，其他所有控件都运行于服务器端，因此控件又分成HTML控件和服务器控件两大类。HTML控件属于客户端（浏览器）控件，服务器无法对其进行控制；服务器端控件可以通过服务器端代码来控制。



图6-1　Visual Studio的控件组

在创建ASP.NET网页时，主要使用以下类型的控件。

✍ HTML控件：HTML控件属于HTML元素，但是若让它运行在服务器端，就变成了HTML服务器控件。

✍ 标准的Web服务器控件：包括最常用的控件，这类控件在网页中占90%，主要是指传统的Web窗体控件。

- 验证控件：这类控件用来验证用户的输入。
- 用户控件：使用创建 ASP. NET 网页的相同技术创建可重复使用的自定义控件。
- 数据控件：这类控件用来链接数据源。该类控件可细分为数据源控件和数据绑定控件两种类型。
- 导航控件：其与站点导航数据结合，实现站点导航功能。我们在任务 3 为在线考试系统制作导航系统时使用的就是该类控件。

ASP. NET 控件具备一些能够简化开发工作的特性，具体如下。

(1) 丰富而一致的对象模型

WebControl 基类实现了对所有控件通用的大量属性，这些属性包括 ForeColor、BackColor、Font、Enabled 等。属性和方法的名称是经过精心挑选的，以此提高在整个框架和该组控件中的一致性。通过这些组件来实现的具有明确类型的对象模型将有助于减少编程错误。

(2) 对浏览器的自动检测

Web 控件能够自动检测客户机浏览器的功能，并相应地调整它们所提交的 HTML，从而充分发挥浏览器的功能。

(3) 数据绑定

在 Web 窗体页面中，可以对控件的任何属性进行数据绑定。此外，还有几种 Web 控件可以用来提交数据源的内容。

6.1.2　HTML 控件

HTML 控件是从 HTML 标记衍生而来的，每个控件对应于一个或一组 HTML 标记，HTML 控件如图 6-2 所示。在默认情况下，HTML 控件属于客户端(浏览器)控件，服务器无法对其进行控制。但是几乎所有的 HTML 标记加上 runat="server"后，就可以变成 HTML 服务器端控件。

HTML 服务器控件派生于 System. Web. UI. HtmlControls. HtmlControl 类，此基类包含所有常用属性。

在文档窗口，只要把这些 HTML 控件中的任何一个从工具箱拖放到 ASP. NET 页面的设计视图或源视图上，就可以生成相应的 HTML 元素。例如，把一个 HTML Button 控件放在页面上，就会在代码中生成如下结果。

```
<input id="Button1" type="button" value="button"/>
```

在此状态下，Button 控件不是一个服务器端控件，而只是一个 HTML 元素。可以用两种不同的方式把它转换为 HTML 服务器控件。

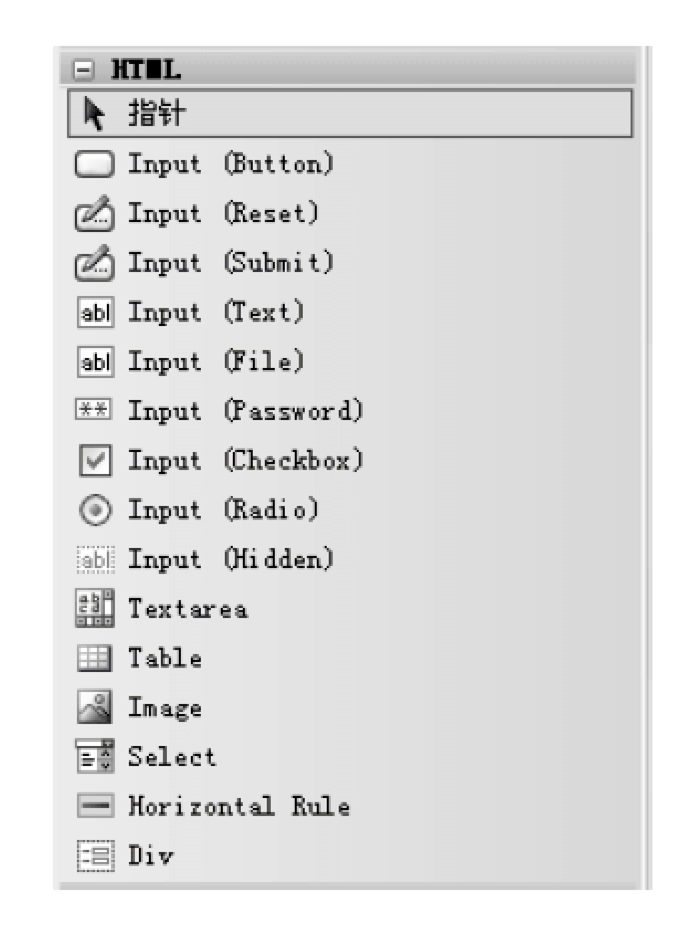


图 6-2　HTML 控件

(1) 在设计视图中，右击该元素，从弹出的快捷菜单中选择“作为服务器控件运行”命令。此时，可视化元素上会出现一个绿色的小三角形。Button 元素转换为 HTML 服务器控件后，如图 6-3 所示。



（2）在源视图中，在相应 HTML 控件上添加 runat="server"，即可把 HTML 控件转换为 HTML 服务器控件：

```
<input id="Button1" type="button" value="button" runat=
"server"/>
```



图 6-3　服务器控件标志

将其转换为服务器控件之后，就可以在服务器端进行编程了。

HTML 服务器控件有一些常用的公共属性，如表 6-1 所示。

表 6-1　HTML 服务器控件的公共属性

属　　性	说　　明
Style	设定控件的样式
Visible	控制控件的显示和消失
Disabled	控制控件是否可用
InnerHtml	以编程方式修改 HTML 服务器控件的开始和结束标记中的内容
InnerText	以编程方式修改 HTML 服务器控件的开始和结束标记之间的内容

下面介绍常用的 HTML 控件的用法。

1．HtmlInput 控件

工具箱的 HTML 选项卡上提供了多个基于 HtmlInput 元素的控件，这些控件分别如下。

- HtmlInput(Text)控件：input type="text"元素
- HtmlInput(Password)控件：input type="password"元素
- HtmlInput(Button)控件：input type="button"元素
- HtmlInput(Submit)控件：input type="submit"元素
- HtmlInput(Reset)控件：input type="reset"元素
- HtmlInput(CheckBox)控件：input type="checkbox"元素
- HtmlInput(Radio)控件：input type="radio"元素
- HtmlInput(File)控件：input type="file"元素
- HtmlInput(Hidden)控件：input type="hidden"元素

HtmlInput 控件有一个重要的属性，即 Type 属性，HtmlInput 控件根据 Type 属性的设定而产生不同种类的输入控件。

小贴士

HtmlInput 控件不需要相对应的结束标记，即只需要撰写<Input 属性"设定值">即可，不需要相对应的</Input>或是以<Input 属性"设定值"/>的方式来作为结束标记。

（1）HtmlInputText 控件

HtmlInputText 控件允许创建单行文本框以接收用户输入，该控件对应于<input type=text>和<input type=password>两种 HTML 元素，此控件在工具箱中的图标为



Input (Text)。与标准 HTML 一样,这些控件可用于在 HTML 窗体中输入用户名和密码。

HtmlInputText 控件常用的属性包括以下几种。

- Value:输入框内的文字。
- Size:设置框的宽度。
- MaxLength:设置可输入的最大字符数。

通过使用 MaxLength、Size 和 Value 属性,可以分别控制可输入的字符数、控件宽度和控件内容。

小贴士

当 Type 属性设置为 password 时,文本框中的输入将受到屏蔽。

(2) HtmlInputButton 控件

可以使用 HtmlInputButton 控件创建按钮控件,该控件分别对应<input type=button>、<input type=submit>和<input type=reset>三种 HTML 元素,允许分别创建命令按钮、提交按钮或重置按钮。用户单击 HtmlInputButton 控件时,来自嵌有该控件的窗体的输入被发送到服务器并得到处理;然后,将响应发送回请求浏览器。这三个按钮控件在工具箱中的图标分别为 Input (Button)、Input (Reset) 和 Input (Submit)。

HtmlInputButton 控件常用的属性和事件如下。

- 属性 Value:设置按钮所显示的文字。
- 事件 Onserverclick。

通过为 ServerClick 事件提供自定义事件处理程序,可以在单击控件时执行特定的指令集。

例 6-1　利用文本输入框取得用户的身份验证信息,用户可以单击 Submit 按钮来确定资料的输入。倘若输入错误的用户名称或密码,则会显示输入错误,提示重新输入。如果单击"清空"按钮,两个输入框会清空(假定用户名和密码都为 asp. NET)。

aspx 页面核心代码如下:

```
输入姓名:<input id="Text1" type="text" runat="server"/><br/>
输入密码:<input id="Password1" type="password" runat="server"/>
<input id="Submit1" type="submit" value="提交" onserverclick="Submit1_
ServerClick" runat="server"/>
<input id="Button1" runat="server" onserverclick="Button1_ServerClick" type="button"
value="清空"/> 
<span id="span1" style="color:Green" runat="server"/></div>
aspx.cs 文件核心代码:
protected void Submit1_ServerClick(object sender, EventArgs e)
{
   if (Text1.Value=="asp.NET" && Password1.Value=="asp.NET")
      span1.InnerHtml="用户名和密码正确";
   else
       span1.InnerHtml="用户名或密码不正确";
}
protected void Button1_ServerClick(object sender, EventArgs e)
{
```



```
    Text1.Value="";
     Password1.Value="";
}
```

用户在文字输入框中所输入的数据会被存在 Value 属性里面，使用者输入完数据后，单击 Submit 按钮则会触发相对应的 OnServerClick 事件程序。我们在事件的程序中检查用户名称及密码是否正确，如果使用者输入正确的用户名称及密码，则会出现输入正确的信息提示，如图 6-4 所示。



图 6-4 运行正确的界面

(3) HtmlInputCheckBox 控件

使用 HtmlInputCheckBox 控件允许创建使用户可以选择 true 或 false 状态的复选框控件，该控件对应于<input type=checkbox> HTML 元素，此控件在工具箱中的图标为 Input (Checkbox) 。

单击 HtmlInputCheckBox 控件时，该控件不会向服务器回送。当使用回送服务器的控件(如 HtmlInputButton 控件)时，复选框的状态被发送到服务器进行处理。若要确定是否选择了复选框，可测试控件的 Checked 属性。

HtmlInputCheckBox 常用的属性如下。

✍ Checked：是否被选取。

✍ Value：获取或设置与 HtmlInputControl 相关联的值。

(4) HtmlInputRadioButton 控件

使用 HtmlInputRadioButton 控件可以创建单选按钮，该控件对应于<input type=radio> HTML 元素，此控件在工具箱中的图标为 Input (Radio) 。如果要将多个单选按钮通过 HtmlInputRadioButton 控件组成一组，可以将 Name 属性设置为组中所有<input type=radio>元素所共有的值。同组中的单选按钮互相排斥，即一次只能选择该组中的一个单选按钮。

HtmlInputRadioButton 控件不会自动向服务器回送，必须依赖于使用某个按钮控件(如 HtmlInputButton、HtmlInputImage 或 HtmlButton)来回送到服务器。可通过为 ServerChange 事件编写处理程序来对 HtmlInputRadioButton 控件进行编程。

HtmlInputRadioButton 控件常用的属性如下。

✍ Checked：是否被选取。



✍ Value：获取或设置与 HtmlInputRadio 相关联的值。

例 6-2　创建 HtmlInputCheckBox 控件来允许用户选择 true 或 false 状态。当用户单击窗口中包含的输入按钮时，Button1_Click 事件处理程序确定是否选中了 HtmlInput-CheckBox 控件。然后，它在<span>控件中显示一个消息。注意，此例中即使在默认情况下将选中的值设置为 true，用户仍然需要单击 Button1 按钮以显示该文本。

Default.aspx 页面关键代码：

```
<body>
    <form id="form1" runat="server">
    <div>
     <h3>HtmlInputCheckBox 示例</h3>
     <input id="Check1" type="checkbox" runat="server" checked NAME="Check1">CheckBox1<
       br/>
     <p>
     <span id="Span1" style="COLOR:red" runat="server"/>
     <p>
     <input type="submit" id="Button1" value="进入" runat="server" NAME="Button1"
       onserverclick="Button1_ServerClick">
    </div>
    </form>
</body>
```

Default.aspx.cs 页面关键代码：

```
protected void Button1_ServerClick(object sender, EventArgs e)
{
   if (Check1.Checked==true)
     Span1.InnerHtml="Check1 被选择!";
   else
       Span1.InnerHtml="Check1 没有被选择!";
}
```

例 6-3　为 HtmlInputRadioButton 控件的 ServerChange 事件创建事件处理程序。此事件处理程序可确定选择哪个单选按钮并将选定内容显示在消息中。

aspx 核心代码：

```
<div>
     <h3>HtmlInputRadioButton 示例</h3>
     <input type="radio" id="Radio1" name="Mode" OnServerChange="Server_Change"
       runat="server" VALUE="Radio1" >选项 1<br>
     <input type="radio" id="Radio2" name="Mode" OnServerChange="Server_Change"
       runat="server" VALUE="Radio2">选项 2<br>
     <input type="radio" id="Radio3" name="Mode" OnServerChange="Server_Change"
       runat="server" VALUE="Radio3">选项 3 <br/>
     <input id="Button1" runat="server" name="Button1" OnserverClick="Button1_
       ServerClick" type="submit" value="进入"/><br/>
     <span id="Span1" runat="server"/>
</div>
```

aspx.cs 文件核心代码：



```
protected void Server_Change(object sender, EventArgs e)
{
    if (Radio1.Checked==true) Span1.InnerHtml="选项 1 被选择";
    else if (Radio2.Checked==true) Span1.InnerHtml="选项 2 被选择";
    else if (Radio3.Checked==true) Span1.InnerHtml="选项 3 被选择";
}
```

运行结果如图 6-5 所示。

（5）HtmlInputFile 控件

HtmlInputFile 控件可以创建一个用于将文件从客户端上传到服务器的控件，该控件对应于<input type=file>Html 标签。该标签将在页面上显示一个文本框和一个用于查找文件的“浏览”按钮。用户可以通过单击该按钮打开“选择文件”对话框来选择文件，选中的文件将显示在文本框中。HtmlInputFile 控件不会自动向服务器回送，必须依赖于使用某个按钮控件（如 HtmlInputButton、HtmlInputImage 或 HtmlButton）来回送到服务器。



图 6-5　HtmlInputRadioButton 控件选择后的界面



图 6-6　成功上传界面

例 6-4　利用 HtmlInputFile 控件上传文件，并保存在 upload 文件夹中，运行结果如图 6-6 所示。

aspx 页面核心代码：

```
<body>
    <form id="form1" runat="server">
    <div>
        <input id="File1" type="file" runat="server"/><br/>
        <input id="Button1" runat="server" type="button" value="上传"
          onserverclick="Button1_ServerClick"/> </div>
        <span id="Span1" runat="server"/>
    </form>
</body>
```

aspx.cs 页面核心代码：

```
protected void Button1_ServerClick(object sender, EventArgs e)
{
  HttpFileCollection _files=System.Web.HttpContext.Current.Request.Files;
                    //得到客户端上传的文件,使用 HttpFileCollection 需添加 using System.IO
  for (int i=0; i<_files.Count; i++)
  {
```



```
        string ShortFileName=_files[i].FileName.Substring(_files[i].FileName.
        LastIndexOf("\\")+1);                          //得到上传文件的名字
        if (ShortFileName!="")
          {
            _files[i].SaveAs(Server.MapPath("~/upload/"+ShortFileName));
                                                       //把文件保存在设定的目录中
          }
        }
        Span1.InnerHtml="上传完毕";
    }
```

利用 HtmlInputFile 控件可以实现类似于邮箱中发送邮件的多文件同时上传效果。

（6）HtmlInputHidden 控件

HtmlInputHidden 控件所创建的不是可视化的控件，该控件对应于<input type=hidden> HTML 标签。尽管此控件是窗体的一部分，但它永远不在窗体上显示。此控件通常与 HtmlInputButton 和 HtmlInputText 控件一起使用。

2．HtmlTable 控件

HtmlTable 控件的主要作用就是用来生成表（Table），该控件在工具箱中的图标为 Table 。可以通过拖动控件的方式将其从工具箱中拖到 Web 页面中，然后利用 Visual Studio 窗口下的“布局”菜单中的命令，实现各种表格操作，包括插入、删除、选定设置表格的效果等。

HtmlTable 控件常用的属性如下。

- Border：设置表格的边框。
- CellPadding：设置单元格之间的距离。
- CellSpacing：设置单元格内的文字与边框之间的距离。
- Width：设置表格的宽度。

3．HtmlImage 控件

HtmlImage 控件用于在 Web 页上显示图片，该控件在工具箱中的图标为 Image 。HtmlImage 控件通常用来显示固定的图片，但是如果让它运行在服务器端，那么可以用编程方式操作 HtmlImage 控件来更改显示的图像、图像大小及图像相对于其他页元素的对齐方式。

HtmlImage 控件的常用属性如下。

- Src：设定需要显示的图像文件。
- Align：对齐图像。
- Alt：设定当图像没有正确加载时，在图像位置显示的文字。
- Border：设定图像边界宽度，当其值为 0 时，表示没有边界。
- Height、Width：设定图像的长、宽值。



图 6-7　单击 Image2 按钮后的效果

例 6-5　当单击 HtmlButton 时以编程方式修改 HtmlImage 控件的属性，运行结果如图 6-7 所示。



aspx 页面核心代码：

```
<form id="form1" runat="server">
    <div>
    <h3>HtmlImage 示例</h3>
    <button id="Button1" OnServerClick="Image1_Click" runat="server" type=
      "button">Image 1</button>
    <button id="Button2" OnServerClick="Image2_Click" runat="server" type=
      "button">Image 2</button><br>
    <img id="Image1" Src="image/image1.jpg" Width="500" Height="226" Alt=
      "Image 1" Border="5" runat="server"/>
    </div>
</form>
```

aspx.cs 页面核心代码：

```
protected void Image1_Click(object sender, EventArgs e)
{
  Image1.Src="image/image1.jpg";
  Image1.Height=100;
  Image1.Width=200;
  Image1.Border=5;
  Image1.Align="center";
  Image1.Alt="图片 1";
}
protected void Image2_Click(object sender, EventArgs e)
{
  Image1.Src="image/image2.jpg";
  Image1.Height=200;
  Image1.Width=300;
  Image1.Border=7;
  Image1.Align="left";
  Image1.Alt="图片 2";
}
```

4．Horizontal Rule 控件

Horizontal Rule 控件即为 HTML 元素中的 HR 标记，此控件用于在 Web 页上显示一条水平分隔线。该控件在工具箱中的图标为 Horizontal Rule 。其使用方法很简单，只需直接把此控件从工具箱拖动到 Web 页面上即可，并且此控件不能作为服务器端控件运行。Horizontal Rule 控件的主要属性就是 Style 属性，用来设置水平分隔线的外观。

6.1.3　标准服务器控件

标准服务器控件包括最常用的控件，这类控件在网页中占 90%，使用它们可以搭建最基本的网页布局。标准服务器控件在 Visual Studio.NET 的控件工具箱中对应的图标如图 6-8

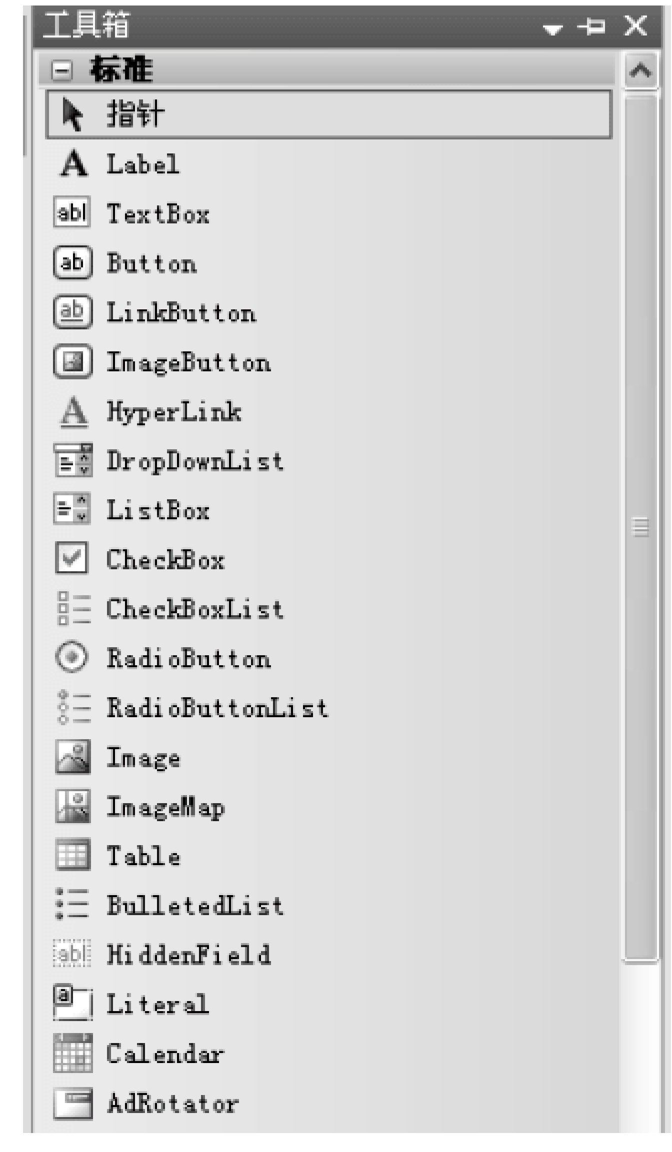


图 6-8　标准服务器控件



所示，使用时可直接将其拖放到 Web 页面上，形成各种布局效果。

下面介绍几种最基本的标准服务器控件。这些标准服务器控件具有一些共同的属性，如表 6-2 所示。

表 6-2　标准服务器控件的公共属性

名　　称	功　　能
AccessKey	获取或设置得以快速导航到 Web 服务器控件的访问键
Attributes	获取与控件的属性不对应的任意特性(只用于呈现)的集合
BackColor	获取或设置 Web 服务器控件的背景色
BorderColor	获取或设置 Web 控件的边框颜色
ForeColor	获取或设置 Web 服务器控件的前景色(通常是文本颜色)
BorderWidth	获取或设置 Web 服务器控件的边框宽度
BorderStyle	获取或设置 Web 服务器控件的边框样式
CssClass	获取或设置由 Web 服务器控件在客户端呈现的级联样式(CSS)表
Style	作为控件的外部标记上的 CSS 样式属性呈现的文本属性集合
Enabled	当此属性设置为 true(默认值)时使控件起作用，设置为 false 时禁用控件
EnableViewState	获取或设置一个值，该值指示服务器控件是否向发出请求的客户端保持自己的视图状态以及它所包含的任何子控件的视图状态
Font	指定 Web 服务器控件的字体属性
Height、Width	获取或设置控件的高度及宽度
SkinID	获取或设置要应用于控件的外观
ToolTip	获取或设置当鼠标指针悬停在 Web 服务器控件上时显示的文本
TabIndex	获取或设置 Web 服务器控件的选项卡索引

1. Label 控件

Label 控件也称作标签控件，可以使用此控件向用户提供基于文本的信息，用户不能编辑或更改这些文本，但是可以通过编程的方式更改其显示的文本或文本的外观。通常，如果只是要显示静态固定的文本，可以使用 HTML 元素直接呈现它，而不需要使用 Label 控件；仅当需要以编程方式更改文本的内容或外观时，才使用 Label 控件。

Label 控件在工具箱中的图标为 A Label，该控件常用的属性如下。

✍ ID：用来更改 Label 控件的 ID 名称。

✍ Text：用来显示初始文本的内容。

2. TextBox 控件

TextBox 控件是让用户输入文本的输入控件，该控件在工具箱中的图标为 abl TextBox。此控件常用的属性和事件如下。



- 属性 Text：字符串，初始化时显示的文字。
- 属性 TextMode：默认情况下，TextMode 属性设置为 SingleLine，它创建只包含一行的文本框。然而，通过将 TextMode 属性值分别改为 MultiLine 或 Password，TextBox 控件也可以显示多行文本框或显示屏蔽用户输入的文本框。
- 属性 MaxLength：表示在文本框中输入的最大字符数。通过设置 MaxLength 属性，可以限制可输入到此控件中的字符数。
- 属性 Rows：当为多行文本时的行数。
- 属性 Columns：控件的宽度。
- 属性 Wrap：表示是否允许自动换行。
- 属性 AutoPostBack：是否允许自动回传事件到服务器。
- 属性 ReadOnly：可以防止控件中显示的文本被修改。
- 事件 OnTextChanged()：当文字改变时触发的事件。

小贴士

文本框的显示宽度(以字符为单位)由它的 Columns 属性确定。如果 TextBox 控件是多行文本框，则它显示的行数由 Rows 属性确定。

3. Button 控件

Button 控件用于显示一个按钮，可以是“提交”按钮或命令按钮。默认情况下，Button 控件是“提交”按钮。“提交”按钮没有与之相关联的命令名(由 CommandName 属性指定)，它只是将网页回发到服务器。可以为 Click 事件提供事件处理程序，以便使用编程方式控制在用户单击“提交”按钮时执行的操作。命令按钮具有命令名，并且允许在一个页面上创建多个命令按钮。可以编写一些事件句柄，在 Command 按钮被单击时来控制动作的执行。

Button 控件在工具箱中的图标为 ab Button，该控件常用的属性和事件如下。

- 属性 ID：Button 控件的名称。
- 属性 CommandName：控件的命令名称。
- 属性 PostBackUrl：单击 Button 控件时从当前页发送到的网页的 URL。默认值为空字符串("")，表示将页回发到自身。
- 事件 OnClick：按钮单击之后执行的操作。
- 事件 OnCommand：当单击 Button 控件时会引发 Command 事件，当命令名与 Button 控件关联时，通常使用该事件。

例 6-6 用户验证程序。

aspx 页面核心代码：

```
<body>
    <form id="form1" runat="server">
    <div>
        用户名：<asp:TextBox ID="txtName" runat="server"></asp:TextBox>
      <br/>
        密码：<asp:TextBox ID="txtPwd" runat="server" TextMode="Password" Width=
        "150px"></asp:TextBox>
```





```
        <br/>
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="提交"/>
        <asp:Button ID="Button2" runat="server" OnClick="Button2_Click" Text="清空"/>
        <asp:Label ID="lblMessage" runat="server" Text="Label" Visible="false">
        </asp:Label></div>
    </form>
</body>
```

aspx.cs 页面核心代码：

```
protected void Button1_Click(object sender, EventArgs e)
{
  if (txtName.Text=="asp.NET" && txtPwd.Text=="asp.NET")
  {
    lblMessage.Visible=true;
    lblMessage.Text=txtName.Text+"用户,你好:";
  }
   else
   {
     lblMessage.Visible=true;
     lblMessage.Text="用户名或密码错误,请重新输入";
    }
}
protected void Button2_Click(object sender, EventArgs e)
{
    txtName.Text="";
    txtPwd.Text="";
}
```

输入正确的用户界面如图 6-9 所示。

输入错误的用户界面如图 6-10 所示。

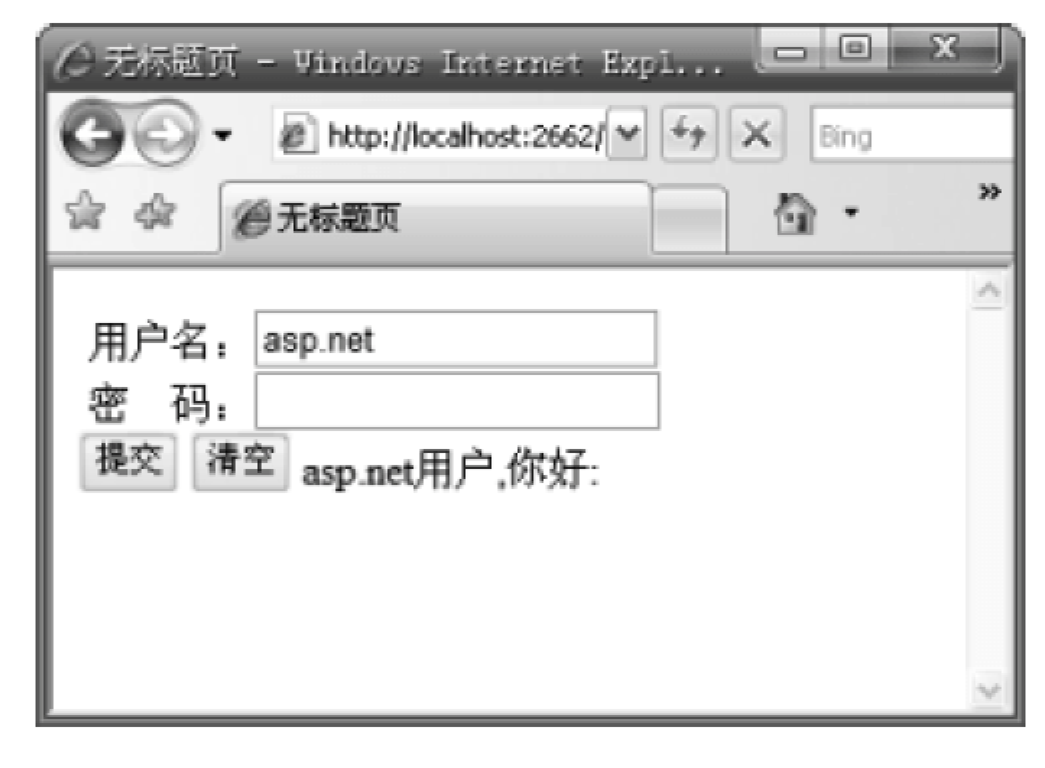


图 6-9　输入正确的用户界面



图 6-10　输入错误的用户界面

4. 选择控件

（1）CheckBox 控件与 CheckBoxList 控件

CheckBox 控件也叫做复选框控件。在日常信息输入中，经常会遇到这样的情况：输入的信息有多种可能性（例如，爱好、兴趣之类），如果采用文本输入的话，一方面是输入烦琐；



另一方面无法对输入信息的有效性进行控制，这时如果采用复选框控件(CheckBox)，就会大大减轻数据输入人员的负担，同时输入数据的规范性也得到了保证。CheckBox控件在工具箱中的图标为☑ CheckBox，此控件常用的属性和事件如下。

✍ 属性ID：控件的名称。

✍ 属性Text：指定要在控件中显示的标题，标题可显示在复选框的右侧或左侧。

✍ 属性Checked：根据其是否为真来判断用户选择与否，复选框允许用户在true或false状态之间切换。

✍ 属性TextAlign：指定标题显示在哪一侧。

✍ 属性AutoPostBack：默认情况下，CheckBox控件在被单击时不会自动向服务器发送窗体。若要启用自动发送，应将该属性设置为true。

✍ 事件CheckedChanged：若要确定是否已选中CheckBox控件，应测试Checked属性。当CheckBox控件的状态在向服务器的各次发送过程中更改时，将引发该事件。

CheckBoxList控件是一个整体式的控件，组成一个可复选的列表项集合，其中每个列表项由一个独立的CheckBox控件生成。子复选框的属性通过读取相关的数据源进行设置。CheckBoxList控件在工具箱中的图标为 CheckBoxList，此控件常用的属性和事件如下。

✍ 属性AutoPostBack：true/false，表示是否自动回传数据。

✍ 属性DataSource：用于指定填充列表控件的数据源。

✍ 属性DataTextField：用于指定DataSource中的一个字段，该字段的值对应于列表项的Text属性。

✍ 属性DataValueField：用于指定DataSource中的一个字段，该字段的值对应于列表项的Value属性。

✍ 属性Items：表示列表中各个选项的集合，如CheckBoxList.Items(i)表示第i个选项，i从0开始。对应每一个Items(i)又可细分为以下属性及方法。

- 属性Text：表示每个选项的文本。
- 属性Value：表示每个选项的选项值。
- 属性Selected：表示该选项是否被选中。
- 方法Count：通过Items.Count方法来获得选项数。
- 方法Add：通过Items.Add方法向RadioButtonList控件添加选项。
- 方法Remove：通过Items.Remove方法，可以从RadioButtonList控件中删除指定选项。
- 方法Insert：通过Items.Remove方法，可将一个新的选项插入到RadioButtonList控件中。
- 方法Clear：通过Items.Clear方法，可以清空RadioButtonList控件中的选项。

✍ 属性RepeatDirection：用于指定显示方向。Vertical是垂直排列，Horizontal是水平排列。

✍ 属性RepeatLayout：用于设置选项的排列方式。属性值为Table时，以表结构显示；否则不以表结构显示。

✍ 属性SelectedIndex：用于获取或设置列表中选定项的最低序号索引值。如果列表中只有一个选项被选中，则该属性表示当前选定项的索引值。

✍ 属性 SelectedItem：用于获取列表控件中索引值最小的选定项。

✍ 属性 TextAlign：用于指定列表中各项文本的显示位置。

✍ 事件 SelectedIndexChanged：当用户选择了列表中的任意选项时，都会触发该事件。

小贴士

由于<asp:CheckBox>元素没有内容，因此可用/>结束该标记，而不必使用单独的结束标记。

当创建多个 CheckBox 控件时，还可以使用 CheckBoxList 控件。对于使用数据绑定创建一组复选框而言，CheckBoxList 控件更易于使用，而各个 CheckBox 控件则可以更好地控制布局。

(2) RadioButton 控件和 RadioButtonList 控件

RadioButton 控件即单选控件，使用单选控件的情况与使用复选控件的条件差不多，区别在于：单选控件的选择可能性不一定是两种，只要是有限种可能性，并且只能从中选择一种结果，原则上都可以用单选控件(RadioButton)来实现。

RadioButton 控件在工具箱中的图标为：⊙ RadioButton，该控件的属性与 CheckBox 控件的属性基本类似，具有 ID 属性、Text 属性，同样也依靠 Checked 属性来判断是否选中，但是与多个复选控件之间互不相关的情况不同，多个单选控件之间存在着联系，要么是同一选择中的条件，要么不是。所以单选控件多了一个 GroupName 属性，它用来指明多个单选控件是否为同一条件下的选择项，GroupName 相同的多个单选控件之间只能有一个被选中。

另外，通过设置 Text 属性指定要在控件中显示的文本，该文本可显示在单选按钮的左侧或右侧。设置 TextAlign 属性来控制该文本显示在哪一侧。如果为每一个 RadioButton 控件指定了相同的 GroupName，则可以将多个单选按钮分为一组。将单选按钮分为一组将只允许从该组中进行互相排斥的选择。

RadioButtonList 控件是一组 RadioButton 控件。当需要在多个项目中做出单一选择时，或需要在程序中改变单选按钮的个数时，使用 RadioButtonList 控件要比使用多个单个的 RadioButton 控件方便。如果要绑定数据源，也必须使用此控件。RadioButtonList 控件更易于使用，而单个 RadioButton 控件则能够更好地控制布局。RadioButtonList 控件与 CheckBoxList 控件的属性、方法和事件基本类似。若要确定 RadioButton 控件是否已选中，可以通过测试 Checked 属性来实现。

5. 列表控件

(1) DropDownList 控件

DropDownList 控件即下拉列表控件，是将选项显示为下拉列表，并从中进行单项选择。它在框中显示选定项，同时还显示下拉按钮，当用户单击此按钮时，将显示选定项的列表。它通常用于显示文本信息，可以静态显示也可以动态显示。DropDownList 控件在工具箱中的图标是 DropDownList，该控件的主要属性和事件如下。

✍ 属性 AutoPostBack：指定在某一项的选择状态发生改变后表单是否被立即提交的一个布尔值。默认值是 false。



✍ 属性 DataSource：使用的数据源。

✍ 属性 DataTextField：数据源中的一个字段，将被显示于下拉列表中。

✍ 属性 DataValueField：数据源中的一个字段，指定下拉列表中每个可选项的值。

✍ 属性 ID：此控件的唯一 ID。

✍ 事件 OnSelectedIndexChanged：当被选项的索引发生改变时将执行的函数的名称。

（2）ListBox 控件

列表框（ListBox）是在一个文本框内提供多个选项供用户选择的控件，它类似于下拉列表，但是没有显示结果的文本框，此控件在工具箱中的图标为 ListBox 。实际中列表框很少使用，大多数情况下都使用列表控件 DropDownList 来代替 ListBox 加文本框的情况。ListBox 控件常用的属性和方法如下。

✍ 属性 SelectionMode：主要是决定控件是否允许多项选择。当其值为 ListSelectionMode. Single 时，表明只允许用户从列表框中选择一个选项；当其值为 List. Selection Mode. Multi 时，用户可以用 Ctrl 键或者 Shift 键结合鼠标，从列表框中选择多个选项。

✍ 属性 DataSource：说明数据的来源可以为数组、列表和数据表。

✍ 属性 AutoPostBack：设定是否自动提交 OnSelectedIndexChanged 事件。

✍ 属性 Items：传回 ListBox Web 控件中 ListItem 的参考。

✍ 属性 Rows：设定 ListBox Web 控件一次要显示的列数。

✍ 属性 SelectedIndex：传回被选取到 ListItem 的 Index 值。

✍ 属性 SelectedItem：传回被选取到 ListItem 的参考，也就是 ListItem 本身。

✍ 属性 SelectedItems：由于 ListBox Web 控件可以复选，被选取的项目会被加入 ListItems 集合中；本属性可以传回 ListItems 集合，只读。

✍ 方法 DataBind：把来自数据源的数据载入列表框的 items 集合。

例 6-7 选择控件和列表控件的使用。

aspx 页面核心代码：

```
<table border="1" cellpadding="0" cellspacing="0" style="width: 475px">
    <tr>
        <td align="center" colspan="2">  用户注册</td>
    </tr>
    <tr>
        <td style="width: 69px">用户名</td>
        <td style="width: 113px">
          <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox></td>
    </tr>
    <tr>
         <td style="width: 69px">性别</td>
         <td style="width: 113px">
<asp:RadioButton ID="RadioButton1" runat="server" GroupName="1" Text="男"/>
<asp:RadioButton ID="RadioButton2" runat="server" GroupName="1" Text="女"/></td>
     </tr>
     <tr>
         <td style="width: 69px">学习类型</td>
```



```
        <td style="width: 113px">
            <asp:DropDownList ID="DropDownList1" runat="server">
            <asp:ListItem>脱产</asp:ListItem>
            <asp:ListItem>定向</asp:ListItem>
            <asp:ListItem>委培</asp:ListItem>
            </asp:DropDownList></td>
</tr>
<tr>
        <td style="width: 69px">年级</td>
        <td style="width: 113px">
            <asp:RadioButtonList ID="RadioButtonList1" runat="server">
            <asp:ListItem Value="0">2007 级</asp:ListItem>
            <asp:ListItem Value="1">2008 级</asp:ListItem>
            <asp:ListItem Value="2">2009 级</asp:ListItem>
            </asp:RadioButtonList></td>
  </tr>
  <tr>
        <td style="width: 69px">个人爱好</td>
        <td style="width: 113px">
        <asp:CheckBox ID="CheckBox1" runat="server" Text="音乐"/> 
        <asp:CheckBox ID="CheckBox2" runat="server" Text="书法"/> 
        <asp:CheckBox ID="CheckBox3" runat="server" Text="绘画"/> 
        <asp:CheckBox ID="CheckBox4" runat="server" Text="体育"/> 
        </td>
    </tr>
    <tr>
          <td style="width: 69px; height: 21px">本学期必修课程</td>
          <td style="width: 113px; height: 21px">
        <asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple">
        <asp:ListItem>数据结构</asp:ListItem>
        <asp:ListItem>微机原理</asp:ListItem>
        <asp:ListItem>软件工程</asp:ListItem>
        </asp:ListBox></td>
    </tr>
    <tr>
         <td style="width: 69px">本学期选修课程</td>
         <td style="width: 113px">
            <asp:CheckBoxList ID="CheckBoxList1" runat="server">
              <asp:ListItem>大学书法</asp:ListItem>
              <asp:ListItem>音乐修养</asp:ListItem>
              <asp:ListItem>西方美术</asp:ListItem>
              <asp:ListItem>古代诗词</asp:ListItem>
              </asp:CheckBoxList></td>
      </tr>
      <tr>
           <td align="center" colspan="2" style="height: 21px">
              <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text=
               "注册"/></td>
      </tr>
   </table>
```



aspx.cs 页面核心代码：

```
protected void Button1_Click(object sender, EventArgs e)
{
    string a1=TextBox1.Text;
    string a2="";
    if (RadioButton1.Checked)
    {   a2="男";   }
    else if (RadioButton2.Checked)
    {   a2="女";   }
    string a6=this.DropDownList1.SelectedItem.Text.ToString();
    string a3=RadioButtonList1.SelectedItem.ToString();
    string a4="";
    if (CheckBox1.Checked)
    {
        a4+=CheckBox1.Text+",";
    }
    if (CheckBox2.Checked)
    {
        a4+=CheckBox2.Text+",";
    }
    if (CheckBox3.Checked)
    {
        a4+=CheckBox3.Text+",";
    }
    if (CheckBox4.Checked)
    {
        a4+=CheckBox4.Text;
    }
    Response.Write("用户名是"+a1+"<br>");
    Response.Write("性别是"+a2+"<br>");
    Response.Write("学习类型是"+a6+"<br>");
    Response.Write("年级是"+a3+"<br>");
    Response.Write("爱好是"+a4+"<br>");
    string a7="";
    for (int i=0; i<ListBox1.Items.Count; i++)
        if (ListBox1.Items[i].Selected)
            a7+=ListBox1.Items[i].Value+"  ";
    string a5="";
    for (int i=0; i<CheckBoxList1.Items.Count; i++)
    {
        if (CheckBoxList1.Items[i].Selected==true)
        {
            a5+=CheckBoxList1.Items[i].Value.ToString()+",";
        }
    }
    Response.Write("必修的课程是"+a7+"<br>");
    Response.Write("选修的课程是"+a5);
}
```

运行效果如图 6-11 所示。





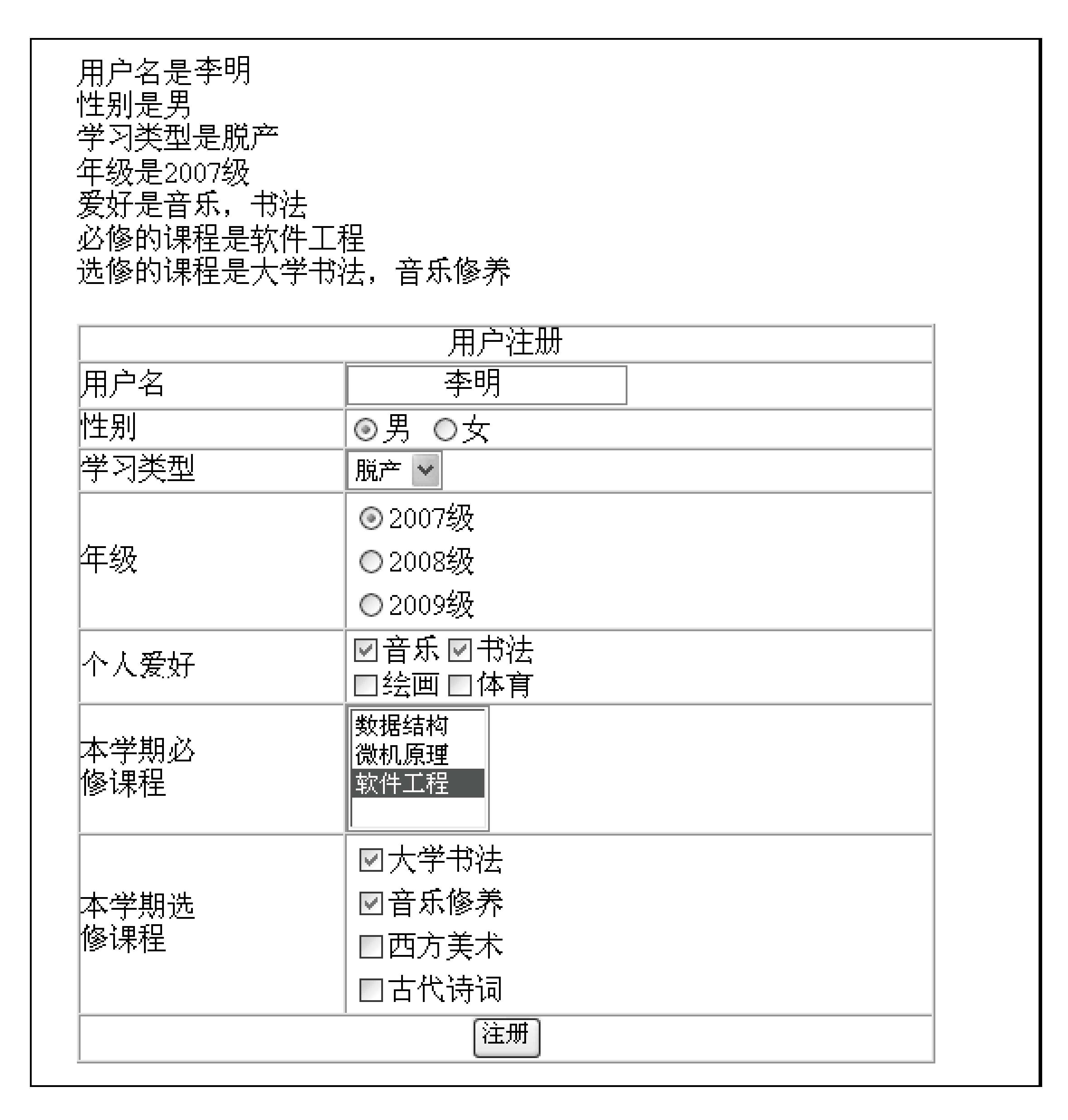


图 6-11　选择控件与列表控件运行效果

如果将 ListBox 的 SelectionMode 属性设为 Multiple，就可以进行多项选择。

6．ImageButton 控件与 Image 控件

ImageButton 控件用于创建图形按钮或用于显示可单击的图像，有很多属性与之前的 Button 相同，可以利用 OnClick 编写单击事件。此控件在工具箱中的图标为 ImageButton，它的常用属性和事件如下。

- 属性 CausesValidation：规定在 ImageButton 控件被单击时，是否验证页面。
- 属性 CommandArgument：有关要执行的命令的附加信息。
- 属性 CommandName：与 Command 事件相关的命令。
- 属性 PostBackUrl：当 ImageButton 被单击时，从当前页面进行回传的目标页面的 URL。
- 事件 OnClick：在单击 ImageButton 控件(单击图像)时发生。
- 事件 OnCommand：在单击 ImageButton 控件时发生。

Image 控件用于显示图像。该控件在工具箱中的图标为 Image，它的常用属性如下。

- AlternateText：图形的替代文本。
- DescriptionUrl：图像进行详细描述的位置。
- ImageAlign：规定图像的排列方式。
- ImageUrl：要使用的图像的 URL。

7．HyperLink 控件与 LinkButton 控件

使用 HyperLink 控件可以在服务器代码中设置链接属性，此外还可以使用数据绑定来指定链接的目标 URL(以及必要时与链接一起传递的参数)。该控件在工具箱中的图标为



A HyperLink，它的常用属性如下。

- ImageUrl：获取或设置为 HyperLink 控件显示的图像的路径。
- NavigateUrl：获取或设置单击 HyperLink 控件时链接到的 URL。
- Target：获取或设置单击 HyperLink 控件时显示链接到的网页内容的目标窗口或框架。
- Text：获取或设置 HyperLink 控件的文本标题。

LinkButton 控件用于在网页上显示超链接样式的按钮控件。它是一种按钮控件，因此具有 OnClick 事件；而 HyperLink 控件只具有链接属性，没有 OnClick 事件。LinkButton 控件在工具箱中的图标为 ab LinkButton，它的常用属性和事件如下。

- 属性 CommandArgument：获取或设置与关联的 CommandName 属性一起传递到 Command 事件处理程序的可选参数。
- 属性 CommandName：获取或设置与 LinkButton 控件关联的命令名。此值与 CommandArgument 属性一起传递到 Command 事件处理程序。
- 属性 PostBackUrl：获取或设置单击 LinkButton 控件时从当前页发送到的网页的 URL。
- 属性 Text：获取或设置显示在 LinkButton 控件上的文本标题。
- 事件 OnClick：在单击 LinkButton 控件时发生。
- 事件 OnCommand：在单击 LinkButton 控件时发生。可以设定 CommandName。

以下代码可以实现超级链接到“搜狐网”。

```
<asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="http://www.sohu.com">
        搜狐网</asp:HyperLink>
<asp:LinkButton ID="LinkButton1" runat="server" OnClick="LinkButton1_Click"
        PostBackUrl="http://www.sohu.com">搜狐网</asp:LinkButton></div>
```

8. Calendar 控件

在实际的网络运用中，时常会需要网页有日历功能，让用户可以很直观地查看当前日期以及通过日历选择日期。

Calendar 控件常用的属性和事件如下。

- 属性 DayNameFormat：指定一周各天的名称格式。
- 属性 FirstDayOfWeek：用于设置日历的最左一列是一周的哪一天。
- 属性 NextMonthText：设置“下一月”导航控件的标题文本。
- 属性 PrevMonthText：设置“上一月”导航控件的标题文本。
- 属性 NextPrevFormat：用于指定 Calendar 控件中“上一月”和“下一月”导航元素的格式。
- 属性 SelectionMode：用于指定用户选择日期的模式。
- 属性 SelectedData：用于获取或设置选定的日期。
- 属性 SelectMonthText：用于设置“月选择器”显示的文本。
- 属性 SelectWeekText：用于设置“周选择器”显示的文本。
- 属性 ShowGridLines：用于设置是否允许在 Calendar 控件中显示格线。

- 属性 ShowNextPrevMonth：设置是否允许在 Calendar 控件的标题中显示“月导航”元素。
- 属性 ShowTitle：用于设置是否显示标题部分。
- 属性 ShowDayHeader：用于设置是否显示星期表头。
- 属性 TodaysDate：用于获取和设置今天的日期。
- 事件 SelectionChanged：当鼠标选中了日期、周或月时会触发。

6.1.4 验证控件基础

1. 验证控件概述

对于交互式网页，用户向服务器提交数据是十分频繁的操作。由于用户也许并不完全了解你的系统，或者手误导致输入了不正确的数据，或者故意输入恶意数据来破坏服务器上的数据库系统，这些行为都可能会对数据库系统带来一定程度的危险。尤其是对于一些特别服务网站(比如银行、证券等)，如果没有一个系统的、严格的验证程序，危险更是无法估计。

验证数据的方式有很多。在以往的 Web 开发中，开发者往往采取手动编码的方式来验证用户输入的数据。这种验证方式需要花费开发者大量的时间，因此他们一直对数据验证比较困惑。除了这种手动编码的验证方式之外，如果开发者熟悉 JavaScript 或者 VBScript，可以用这些脚本语言轻松实现验证，但是需要考虑用户浏览器是否支持这些脚本语言；如果对这些语言工具不是很熟悉或者想支持所有用户浏览器，就必须在具体的程序里面进行手动编码的验证，这种验证就会增加服务器负担。为此，ASP. NET 提供了一系列的验证控件，利用这些验证控件，程序员不仅可以轻松实现对用户输入的验证，而且，还可以选择验证在服务器端进行还是在客户端进行。因此，程序员可以将主要精力放在主程序的设计上。



图 6-12 验证控件

验证控件在工具箱中的位置如图 6-12 所示。

ASP. NET 提供了 5 个验证控件和 1 个汇总控件，5 个验证控件可以实现不同的验证功能。这 6 个控件分别如下。

- RequiredFieldValidator：用于监视控件必须填有数据。
- RangeValidator：用于输入值范围限制。
- RegularExpressionValidator：用于正则表达式验证。
- CompareValidator：用于比较两个监视控件的值。
- CustomValidator：允许用户自编写验证函数。
- ValidationSummary：用于收集显示错误信息。

2. RequiredFieldValidator 控件

RequiredFieldValidator 是一个简单实用的控件，它的主要用处是验证其监视的控件中



是否输入了内容(除空白符和空格之外,其他任意字符均视为有效输入),也称之为非空验证控件。RequiredFieldValidator 控件在工具箱中的图标为 RequiredFieldValidator ,该控件的主要属性如下。

✍ ControlToValidate:表示要进行验证监视的控件。

✍ Text:出错时的提示。

✍ ErrorMessage:提交给 ValidationSummary 的错误提示。当 Text 为空时,也会显示该错误信息。

✍ Display:设置来自多个验证控件的错误信息的空间排列方式。有 3 个取值:Static 表示为错误信息定义固定的布局,即使没有错误信息文本,验证控件也将占用空间;Dymatic 表示使验证控件作为文本流的一部分呈现在页面上,即当验证控件没有被触发时,页面上就不会显示空格;None 表示验证信息从不显示。

3. CompareValidator 控件

CompareValidator 控件用来比较两个控件的输入是否符合程序设定,即是否符合指定的值或者是否符合另外一个输入控件的值,也称之为比较控件。这里的符合,不仅仅表示为"相等",尽管相等是应用得最多的,其实,这里的比较包括范围很广。CompareValidator 控件在工具箱中的图标为 CompareValidator ,该控件的主要属性如下。

✍ ControlToValidate:要验证的控件 ID。

✍ ErrorMessage:出错时提示的错误信息。

✍ ControlToCompare:要比较的控件 ID。

✍ Type:用于指定比较的值的数据类型。

✍ Operator:用于指定要使用的比较运算符。

✍ Display:设置来自多个验证控件的错误信息的空间排列方式。

4. RangeValidator 控件

RangeValidator 控件用于验证用户文本框输入的内容是否在设定的范围之内。比如,要求输入年龄的时候,输入值要在 0～120 内(一般情况下)。此外,该控件不仅仅是限于验证输入的数值,还可以验证输入的字母。比如我们设定 MaximumValue = x,MinimumValue=d,这表示输入的值在字母顺序表中的 d～x 范围内是可以接受的,但是,当输入的值是在 a～c 或是 y、z 范围内,就会提示出错。RangeValidator 控件在工具箱中的图标为 RangeValidator ,该控件的主要属性如下。

✍ ControlToValidate:表示要验证的控件 ID。

✍ MaximumValue:表示控制范围的最大值。

✍ MinimumValue:表示控制范围的最小值。

✍ ErrorMessage:表示当验证的控件输入超出范围时的提示信息。

5. RegularExpressionValidator 控件

RegularExpressionValidator 控件是一个字符串验证控件,因为结合了 RegularExpression(正则表达式),使其成为.NET 平台下最强大的字符串验证控件。该控件通过把用户输入



的字符、数字和符号的模式与控件中的一个或多个模式相比较，来验证用户的输入是否匹配预定义的模式(比如电话号码、邮政编码、电子邮件地址等)。该控件在工具箱中的图标为 RegularExpressionValidator，其主要属性如下。

- ControlToValidate：表示要进行验证的控件 ID。
- ErrorMessage：表示出错提示信息。
- ValidationExpression：表示验证表达式。例如系统默认的邮件地址表达式：

```
ValidationExpression="\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
```

6. CustomValidator 控件

CustomValidator 控件用于用户自定义验证的方式，这种方式是当前面几种验证都不能满足验证需求的时候，程序员可以通过此控件自己定制验证逻辑，针对变量、公式或者其他来源的输入执行检验。该控件可以手写 JSpript 脚本进行客户端验证，也可以编写服务器端的验证事件，或者联合两种方式验证，实现更安全的验证。CustomValidator 控件在工具箱中的图标为 CustomValidator，该控件的主要属性如下。

- ControlToValidate：要验证的控件 ID。
- ClientValidationFunction：用于客户端验证的函数。
- OnServerValidate：服务器端验证的事件方法。
- ErrorMessage：表示出错提示信息。

7. ValidationSummary 控件

ValidationSummary 控件收集本页的所有验证错误信息，并可以将它们组织以后再显示出来。当使用前几个验证控件的属性时，如果 ErrorMessage 和 Text 属性同时设置，出现错误时，错误位置只会显示 Text 的内容。这时，如果同时还要收集 ErrorMessage 中的错误信息，就可以使用 ValidationSummary 控件。该控件的主要属性如下。

- ShowMessageBox：指示是否显示弹出的提示消息(true/false)。
- ShowSummary：指示是否显示该报告内容。
- DisplayMode：表示错误信息显示方式。List 相当于 HTML 中的
，BulletList 相当于 HTML 中的<li>，SingleParegraph 表示错误信息之间不作任何分割。

小贴士

ShowMessageBox 和 ShowSummary 属性不能同时设置；此外，为避免多处提示验证错误信息，可将验证控件的 Text 属性设置为“ * ”号。

6.2 任 务 实 施

6.2.1 创建管理员用户登录页面

当管理员登录在线考试系统的后台管理系统时，必须在用户登录界面输入用户名和密码以确定是否有权限进行管理。如图 6-13 所示，页面由一些文本框和按钮构成，利用验证



控件实现对用户输入的验证。



图 6-13 登录系统

(1) 在已经建立的三层架构的页面表现层上右击，在弹出的快捷菜单中选择“新建”|“文件夹”命令，设置文件夹的名称为 admin，将来存放后台管理员页面。

(2) 右击 admin 文件夹，在弹出的快捷菜单中选择“添加新项”命令，并在打开的对话框中，选择“Web 窗体”选项，并在下面的“名称”文本框中修改网页的名称为“AdminLogin.aspx”，单击“添加”按钮，即可添加一个新的页面。

(3) 在页面中插入一个 6 行 3 列的表格，并将第 1、4、5、6 行合并单元格；在表格第 1 行插入一个“欢迎登录在线考试系统”的图片，第 2 行输入文字“用户名”，第 3 行输入文字“密码”，第 4 行插入一条水平分隔线，完成页面的初始布局，如图 6-14 所示。

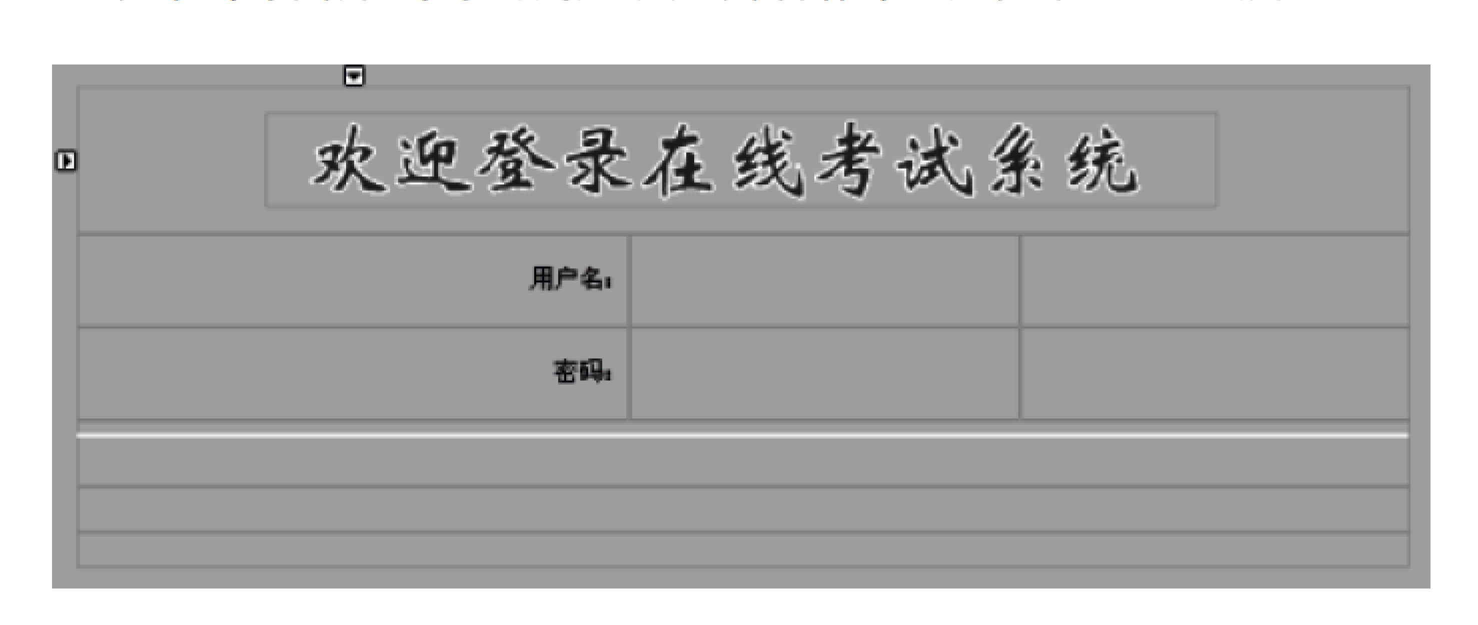


图 6-14 登录界面的初始布局

小贴士

前面的关于整个页面的布局可以用 Dreamweaver 工具去做，之后把代码粘贴到 aspx 页面中。

(4) 为了实现用户名和密码的输入，需要添加两个文本框。从标准服务器控件栏向第 2 行第 2 列的单元格中拖入一个 TextBox 文本框，并设置其 ID 属性值为 txtName，同样向第 3 行第 2 列的单元格中拖入一个 TextBox 文本框，并设置其 ID 属性值为 txtPwd，TextMode 属性值为 Password。向第 5 行的单元格中拖入两个 Button 按钮，并设置这两个按钮的 ID 分别为 btnAdminLogin 和 btnReset，Text 属性分别为“进入系统后台”和“重置”，这些组件设置后的效果如图 6-15 所示。

(5) 运行 AdminLogin.aspx 页面，效果如图 6-13 所示。





图 6-15　完整的登录界面

6.2.2　用户登录的实现

当单击“进入系统后台”按钮时，应该进行下列处理。

第 1 步，从后台数据库中查找该用户名是否存在，如果存在返回用户对象，如果不存在则返回空对象。

第 2 步，如果返回的是用户对象，比较该对象的密码和输入的密码是否一致，如果一致返回 true 值，如果不一致返回 false 值，如果返回的是空对象则返回 false 值。通过前两个步骤，如果用户名和密码正确，得到 true 值；如果用户名和密码有一个不正确，得到“false”值。

第 3 步，根据第二步得到的结果，如果是 true，页面跳转到用户请求的页面或默认页面；如果得到的是 false，向用户提示“用户名或密码错误!”。

以上三步根据功能和处理的对象，分别在数据处理层、逻辑处理层和表示层实现。

1. 数据处理层

```
//根据登录名获取信息
public static examAdmin GetAdminByName(string adminName)
{
    string sql="SELECT * FROM exam_admin WHERE admin_name=@AdminName";

    try
    {
        SqlDataReader reader=ConnDBHelper.GetReader(sql, new SqlParameter("@AdminName",
        adminName));
        if (reader.Read())
        {
            examAdmin exam_Admin=new examAdmin();

            exam_Admin.Admin_id=(int)reader["admin_id"];
            exam_Admin.Admin_name=(string)reader["admin_name"];
            exam_Admin.Admin_pass=(string)reader["admin_pass"];
            exam_Admin.Mgquestions=(int)reader["mgquestions"];
            exam_Admin.MgStudents=(int)reader["mgStudents"];
            exam_Admin.MgSystem=(int)reader["mgSystem"];

            reader.Close();
```



```
                return exam_Admin;
            }
            else
            {
                reader.Close();
                return null;
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            throw e;
        }
}
```

2. 逻辑处理层

```
//管理员登录
public static bool AdminLogin(string adminName, string adminPwd)
{
    examAdmin admin=AdminService.GetAdminByName(adminName);

    //用户名不存在
    if (admin==null)
    {
       return false;
    }

    if (admin.Admin_pass==adminPwd)
    {
        return true;
    }
    else
    {
        //密码错误
        return false;
    }
}
```

3. 表示层

双击“进入系统后台”按钮，为其添加 OnClick 事件，在该事件中添加如下代码。

```
protected void btnAdminLogin_Click(object sender, EventArgs e)
{
    string strRedirect;
        //表单验证所指定的路径
        strRedirect=Request["ReturnUrl"];
```



```
            //表单验证
            System.Web.Security.FormsAuthentication.SetAuthCookie(this.txtName.Text, true);
        if (AdminManager.AdminLogin(this.txtName.Text,this.txtPwd.Text))
        {

            Response.Redirect("~/admin/ListAllStudents.aspx");
            Response.Redirect(strRedirect);
        }
        else
        {
          Response.Write("<script>alert('用户名或密码错误!');</script>");
        }
    }
```

“重置”的实现比较简单,方法为双击“重置”按钮,为其添加 OnClick 事件,在该事件中添加如下代码。

```
    protected void btnReset_Click(object sender, EventArgs e)
    {
        this.txtName.Text=String.Empty;
        this.txtPwd.Text=String.Empty;
    }
```

6.2.3　使用验证控件完善管理员登录功能

为了避免用户忘记输入用户名或密码,需要对用户名和密码的文本框执行非空验证。从验证控件栏向表格第 2 行第 3 列的单元格中拖入 RequiredFieldValidator 控件,设置其 ID 属性的值为 rfvUserName,ControlToValidate 属性的值为 txtName,ErrorMessage 属性的值为“请输入用户名”;同样向表格第 3 行第 3 列的单元格中拖入 RequiredFieldValidator 控件,设置其 ID 属性的值为 fvPd,ControlToValidate 属性的值为 txtPwd,ErrorMessage 属性的值为“请输入密码”。完成添加了验证控件的登录界面如图 6-16 所示。



图 6-16　添加了验证控件的登录界面

运行布局好的 AdminLogin. aspx 页面,当用户名和密码为空时,验证控件会给出提示。



练　　习

1. 单项选择题

（1）HtmlInput 控件不包括________。

A. HtmlTable　　B. HtmlText

C. HtmlCheckBox　　D. HtmlFile

（2）TextBox 控件的 TextMode 属性设置不包括________。

A. SingleLine　　B. MultiLine

C. Password　　D. Line

（3）通过________属性可以判定 CheckBox 控件是否被选中。

A. Checked　　B. Enabled

C. Disabled　　D. Selected

（4）以下控件中可以实现下拉列表的是________。

A. TextBox　　B. DropDownList

C. CheckBoxList　　D. RadioButtonList

（5）可使用________将用户的输入与某个常数值或其他控件的值进行比较。

A. RequiredFieldValidator　　B. CompareValidator

C. RangeValidator　　D. CustomValidator

2. 简答题

（1）列举几种常用的 HTML 控件。

（2）列举几种常用的标准服务器控件。

（3）简述验证控件的作用。

实　　训

实训目的

（1）熟练掌握 ASP. NET 的 HTML 控件和标准服务器控件的使用。

（2）熟练掌握 ASP. NET 验证控件的使用。

实训内容

（1）实现在线考试系统客户端验证功能。

（2）新建一个网站 Exercise 6，实现一个完整的用户注册程序，如图 6-17 所示。提交时，验证各个输入控件的值，提交成功之后提示“用户注册成功”。

实训指导

（1）代码参考在线考试系统。







图 6-17　用户注册程序

（2）尝试实现考生用户注册程序。（提示：此时需要用 ADO. NET 对象连接数据库）

任务7　ASP.NET内置对象与登录页面完善

技能目标

掌握 Response 对象页面导向和文件写入技术，掌握 Request 对象页面传值并调用 Request 对象各种属性的方法，掌握 Session 对象在页面之间实现传值功能。

知识目标

掌握 Response、Request、Application、Session 和 Server 等对象的常用属性和方法。

任务描述

本部分对 ASP. NET 的一些内置对象进行了介绍，并进一步完善了登录页面。

本章任务如下：

- 掌握 Application 对象实现网上在线人数统计；
- 完善在线考试系统后台管理登录页面；
- 实现网页浏览计数器。

7.1　知识准备

7.1.1　ASP. NET 内置对象概述

在 ASP. NET 中包含一系列类，在页面中可以直接使用，我们称之为内置对象。ASP. NET 的内置对象主要包括 Page、Request、Response、Application、Session、Server 和 Cookie 等。其中 Application、Session 和 Cookie 都能够用来存储应用程序的数据，但它们之间又有所不同。其中 Application 对象被整个应用程序所共享，即多个用户共享同一个 Application 对象，经常用来存储整个应用程序相关的数据；Session 对象被每一个用户所独享，且每一个用户都具有唯一的 Session 标识（Session 对象的 ID），经常用来存储用户相关数据；Cookie 对象可用于保存客户端浏览器请求的服务器页面，也可以用它存放非敏感性的用户信息，信息的保存时间可以根据用户需要进行设置，但 Cookie 和其他对象的最大不同是 Cookie 将信息保存在客户端，而 Session 和 Application 是保存在服务器端。也就是说，无论何时用户连接到服务器，Web 站点都可以访问 Cookie 信息。这样，既方便用户的

使用，也方便了网站对用户的管理。Request 对象是 ASP. NET 的重要对象，它可以用于服务器端和客户端交换数据，表示用客户端向服务器端发送的 HTTP 请求。Request 对象用于服务器端从客户端获得数据；而 Response 对象主要用于服务器端向客户端输出数据，实现页面跳转等功能，还可以用来传递各个页面之间的参数。它和 Response 对象共同实现了服务器和客户端的交互，以及数据的交互等功能。Server 对象提供了访问服务器对象的方法和属性，可以获取服务器信息，如应用程序的物理路径等。

7.1.2　Page 对象

Page 对象代表. aspx 文件。了解 Page 对象对于灵活控制 ASP. NET 的基本形态是十分必要的。

1. @Page 指令

在 Visual Studio 的窗口中，新建页面的. aspx 文件有两种视图：设计视图和源视图。打开源视图，在第一行可以看到这样一行代码：

```
<%@ Page   Language="C#"   AutoEventWireup="true"   CodeFile="Default.aspx.cs"
Inherits="_Default" %>
```

其中，

Language="C＃"：指明页面代码和后置代码所使用的脚本语言。

AutoEventWireup="true"：设置是否自动调用网页事件，默认为 true。

CodeFile="Default. aspx. cs"：指定代码后置文件，该文件包含此页面的局部类。

Inherits="_Default"：代码后置的类，局部类。

@Page 指令定义了 ASP. NET 页用于编译和解析的属性。每个 aspx 页面只能有一个@Page 指令。

2. Page 对象的常用属性

(1) IsPostBack 属性

IsPostBack 属性用于判断页面是回传或者是首次加载。

IsPostBack 常放置于 Page_Load 方法中，用于首次加载页面时的初始化。

例如：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Page.IsPostBack)
    {
        …
    }
}
```

(2) PreviousPage 属性

PreviousPage 属性用于获取向当前页面传输数据的页面。执行跨页数据传送时会用到该属性。



3. Page 对象的常用方法

Page_Load 方法：加载页面时触发该页面，如前例。

7.1.3 Request 对象

1. Request 对象的常用属性

(1) Browser 属性

Browser 属性用于获取或设置有关正在请求的客户端浏览器的信息。

例如：

```
Label1.Text="浏览器使用的平台："+Request.Browser.Platform+"<br>"+"浏览器类型："+
        Request.Browser.Type+"<br>"+
        "浏览器版本："+Request.Browser.Version+"<br>";
```

(2) QueryString 属性

用于获取 HTTP 查询字符串变量的集合。

例如：

```
Response.Redirect("welcome.aspx?name="+yhm.Text);
      //跳转到 welcome.aspx 页面在 welcome.aspx 页面中获得传递过来的用户名并输出到页面上
Response.Write (Request.QueryString ["name"]+"您好!欢迎您的光临!<br>");
```

2. Request 对象的常用方法

SaveAs 方法：用于将 HTTP 请求保存到硬盘上，在调试过程中非常有用。例如，在应用程序中调用 Request 对象的 SaveAs 方法，将 HTTP 请求保存到本网站虚拟目录下的 test.txt 文件中。

例如：

```
Request.SaveAs (Server.MapPath ("test.txt"),true);
```

Request 对象的常用属性如表 7-1 所示。

表 7-1　Request 对象的常用属性

属　　性	功 能 说 明
Cookies	获取相应的 Cookie 集合
Files	获取或设置该页在浏览器上缓存过期之前的分钟数
Form	获取一个值，该值指示客户端是否仍连接在服务器上
QueryString	清除缓冲区中的所有内容输出
ServerVariables	刷新缓冲区，向客户端发送当前所有缓冲的输出
UserHostAddress	将当前所有缓冲的输出发送到客户端，停止该页的执行



7.1.4　Response 对象

1. Response 对象的常用属性

(1) ContentType 属性

ContentType 属性用来获取或设置输出流的 HTTP MIME 类型，也就是 Response 对象向浏览器输出内容的类型，默认值为 text/html。ContentType 的值为字符串类型，格式为 type/subtype，type 表示内容的分类，subtype 表示特定内容的子类。

例如：

```
Response.ContentType="image/gif";
```

表示向浏览器输出的内容为 gif 图片。

(2) IsClientConnected 属性

IsClientConnected 属性用于获取一个值，通过该值指示客户端是否仍连接在服务器上。

例如：

```
if(Response.IsClientConnected)
    Response.Redirect("welcome.aspx",false);
else
    …
```

2. Response 对象的常用方法

(1) Redirect 方法

Redirect 方法将网页重新导向另一个地址。例如用户从登录页面进入网站主页时，输入合法的用户名和密码之后，通过调用 Response 对象的 Redirect 方法就可以跳转到网站的主页面。

例如：

```
Response.Redirect("welcome.aspx");        //跳转到 welcome.aspx 页面不传递数据
Response.Redirect("welcome.aspx?name="+yhm.Text);
                    //跳转到 welcome.aspx 页面的同时传递数据，yhm 是当前页面 TextBox 的 ID
```

(2) Write 方法

Write 方法用来将数据输出到客户端。

例如：

```
Response.Write ("您好!欢迎您的光临!<br>");
Response.Write (Request.QueryString ["name"]+"您好!欢迎您的光临!<br>");
```

(3) WriteFile 方法

WriteFile 方法将指定的文件直接写入 HTTP 内容输出流。

例如：

```
Response.WriteFile("aa.txt");              //aa 是一个文本文件
```



Response 对象的常用属性和方法如表 7-2 所示。

表 7-2 Response 对象的常用属性和方法

名　称	功能说明
BufferOutput 属性	获取或设置一个值,该值指示是否缓冲输出
Cookies 属性	获取相应 Cookie 集合
Expires 属性	获取或设置该页在浏览器上缓存过期之前的分钟数
Clear 方法	清除缓冲区中的所有内容输出
Flush 方法	刷新缓冲区,向客户端发送当前所有缓冲的输出
End 方法	将当前所有缓冲的输出发送到客户端,停止该页的执行

7.1.5 Application 对象

Application 对象可称为记录应用程序参数的对象。有时编程人员希望能存储一段信息,这段信息可能被整个网站的所有页面使用,这时就需要使用 Application 对象。每个 Application 对象变量都是 Application 集合中的对象之一,由 Application 对象统一管理。由于多个用户可以共享一个 Application 对象,所以必须使用 Lock 和 Unlock 方法,以确保多个用户无法同时改变某一个属性。Application 对象变量的生命周期中止于关闭 IIS 或使用 Clear 方法清除,Clear 方法是 Page 对象的成员,可以直接调用。

说明:一个网站可以有不止一个 Application 对象。典型情况下,可以针对个别任务的一些文件创建个别的 Application 对象。例如,可以建立一个 Application 对象来适用全部公用用户,而再创建另外一个只适用于网络管理员的 Application 对象。

1. Application 对象的常用属性

(1) AllKeys 属性

AllKeys 属性用于返回全部 Application 对象变量名到一个字符串数组中。例如,使用 Application 对象 Item 属性的两种重载格式访问 Application 对象集合中的项,并输出到 Web 页面中。

例如:

```
Application["ap1"]="app1";
Application["ap2"]="app2";
Application["ap3"]="app3";
Response.Write (Application["ap1"].ToString );
Response.Write ("</br>");
Response.Write (Application[2].ToString());
```

(2) Count 属性

Count 属性用于获取 Application 对象变量的数量。

例如:

```
Application["ap1"]="app1";
```



```
Application["ap2"]="app2";
Application["ap3"]="app3";
Response.Write ("Application 对象数量为: "+Application.Count.ToString()+
"个,分别为: <br>"+Application["ap1"]+","+Application["ap2"]+
","+Application["ap3"]);
```

（3）Item 属性

Item 属性有两种格式，可以是通过索引获取单个 Application 对象的值，还可以是通过名称获取单个 Application 对象的值。

例如：

```
Application["ap1"]="app1";
Application["ap2"]="app2";
Application["ap3"]="app3";
Response.Write (Application["ap1"].ToString );
Response.Write ("</br>");
Response.Write (Application[2].ToString());
```

2. Application 对象的常用方法

Lock 方法和 Unlock 方法：用于对全部 Application 对象变量进行锁定和解锁。例如，统计在线人数时，就应该先对 Application 对象加锁，以防止因为多个用户同时访问页面而造成并行。

例如：

```
Application.Lock ();
Application["count"]= (int)Application["count"]+1;
Application.UnLock ();
```

3. Application 对象的事件

Application 对象有 4 个事件，分别介绍如下。

（1）OnStart 事件：在整个 ASP. NET 应用中首先被触发的事件，也就是在一个虚拟目录中第一个 ASP. NET 程序执行时触发。

（2）OnEnd 事件：在整个 ASP. NET 应用停止时被触发。

（3）OnBeginResquest 事件：在每一个 ASP. NET 程序被请求时就发生，即客户每访问一个 ASP. NET 程序时，就触发一次该事件。

（4）OnEndResquest 事件：ASP. NET 程序结束时，触发该事件。

Application 对象的常用属性和方法如表 7-3 所示。

表 7-3　Application 对象的常用属性和方法

名　　称	功 能 说 明
AllKeys 属性	返回全部 Application 对象变量名到一个字符串数组中
Count 属性	获取 Application 对象变量的数量
Item 属性	允许使用索引或 Application 变量名称传回内容值



续表

名　　称	功 能 说 明
Add 方法	新增一个 Application 对象变量
Clear 方法	清除全部 Application 对象变量
Lock 方法	锁定全部 Application 对象变量
Remove 方法	使用变量名称移除一个 Application 对象变量
RemoveAll 方法	移除全部 Application 对象变量
Set 方法	使用变量名称更新一个 Application 对象变量的内容
UnLock 方法	解除锁定的 Application 对象变量

7.1.6　Session 对象

Session 对象可称为记录浏览器端的变量对象。Session 是用来存储跨网页程序的变量或者对象，Session 对象只针对单一网页使用者，也就是说各个连接的机器都有各自的 Session 对象，不同的客户端无法互相存取。Session 对象中止于联机机器离线时，也就是当网页使用者关掉浏览器或超过设定 Session 变量的有效时间时，Session 对象就会消失。

1. Session 对象的常用属性

(1) Item 属性

Item 属性有两种格式：可以按数字索引获取或设置会话值，也可以按名称获取或设置会话值。

(2) TimeOut 属性

TimeOut 属性获取并设置在会话状态提供程序终止会话之前各请求之间所允许的时间(以分钟为单位)。在应用程序开发过程中，要更改 Session 对象的有效期限，只要设定 TimeOut 属性即可，TimeOut 属性的默认值是 20 分钟。

例如：启动 Visual Studio. NET 2005，创建一个新的项目，默认主页名为 Default. aspx。在 Web 窗体中拖放 3 个 Label 控件；一个 Button 控件，其 Text 属性为 OK。双击 OK 按钮，添加如下代码。

```
private void Button1_Click(object sender, System.EventArgs e)
{
    DateTime dt=DateTime.Now ;
    Label1.Text=dt.ToString();
    Label2.Text=Session["session1"].ToString();
    Label3.Text=Session["session2"].ToString();
}
```

在 Page_Load 方法中输入以下代码。

```
private void Page_Load(object sender, System.EventArgs e)
```



```
{
    if(!IsPostBack)
    {
        Session["session1"]="value1";
        Session["session2"]="value2";
        Session.Timeout=1;
        DateTime dt=DateTime.Now ;
        Label1.Text=dt.ToString();
        Label2.Text=Session["session1"].ToString();
        Label3.Text=Session["session2"].ToString();
    }
}
```

编译执行代码，结果如图 7-1 所示。等待一分钟后，单击 OK 按钮，会发现页面报错，这时因为两个 Session 变量的时间已经超时了，是 Session 的值无法赋给 Label 控件的原因造成的。



图 7-1　Default. aspx 执行结果

2. Session 对象的常用方法

CopyTo 方法：将会话状态值的集合复制到一维数组中(从数组的指定索引处开始)。

例如：启动 Visual Studio. NET 2005，创建一个新的项目，默认主页名为 Default. aspx。在 Page_Load 方法中输入以下代码。

```
private void Page_Load(object sender, System.EventArgs e)
{
    Session["sess1"]="value1";
    Session["sess2"]="value2";
    Session["sess3"]="value3";
    string[] strArray=new string[] {"1","2","3","4"};
    Response.Write("原数组如下：</br>");
    foreach(string str in strArray)
    {
        Response.Write (str+"</br>");
    }
    Session.CopyTo(strArray,0);
```



```
        Response.Write("新数组如下：</br>");
      foreach(string str in strArray)
      {
          Response.Write (str+"</br>");
      }
}
```

编译执行代码，结果如图 7-2 所示。



图 7-2　应用 CopyTo 方法的执行结果

3. Session 对象的事件

(1) OnStart 事件：在客户第一次从应用程序中请求 ASP. NET 网页时由 ASP. NET 调用。

(2) OnEnd 事件：在会话关闭时调用。

Session 对象的常用属性和方法如表 7-4 所示。

表 7-4　Session 对象的常用属性和方法

名　　称	功能说明
Contents 属性	获取对当前会话状态对象的引用
Item 属性	获取或设置会话值
TimeOut 属性	传回或设定 Session 对象变量的有效时间，当使用者超过有效时间没有动作，Session 对象就会失效。默认值为 20 分钟
Abandon 方法	此方法结束当前会话，并清除会话中的所有信息。如果用户随后访问页面，可以为它创建新会话（“重新建立”非常有用，这样用户就可以得到新的会话）
Add 方法	向 Session 对象集合中添加一个新项
CopyTo 方法	将会话状态值的集合复制到一维数组中（从数组的指定索引处开始）
Clear 方法	此方法清除全部的 Session 对象变量，但不结束会话





7.1.7　Server 对象

1. Server 对象的常用属性

（1）MachineName 属性

MachineName 属性用于获取服务器的计算机名称。

例如：

```
Label1.Text="本计算机名为："+Server.MachineName.ToLower ();
```

（2）ScriptTimeout 属性

ScriptTimeout 属性用于获取和设置请求超时时间，默认时间为 90 秒。如果一个文件执行时间超过此属性设置的时间，则自动停止执行，这样可以防止某些可能进入死循环的程序导致服务器资源的大量消耗。如果页面需要较长的运行时间，比如要上传一个非常大的文件，就需要设置一个较长的请求超时时间。

例如：

```
Server.ScriptTimeout=200;
```

2. Server 对象的常用属性

（1）HtmlEncode 方法

当字符串中包含有 HTML 标记时，浏览器会根据标记的作用来显示内容，而标记本身并不显示在页面上。但有时需要在页面上显示 HTML 语句，比如要在页面上显示一个网页的源文件等。另外，在留言本、论坛等程序中，如果用户输入的信息中包含有 HTML 代码或一些客户端脚本程序等，则可能会对网站造成一定的危害。Server 对象的 HtmlEncode 方法就是用来将字符串中的 HTML 标记字符转换为字符实体，从而使 HTML 标记本身显示在页面上。

例如：

```
string str1,str2;
str1="<h2>大家好!</h2>";
str2=Server.HtmlEncode(str1);
Response.Write(str1);
Response.Write(str2);
```

执行结果如图 7-3 所示。

（2）UrlEncode 方法

Server 对象的 UrlEncode 方法，是用来对字符串进行 URL 格式编码的。在 URL 中，有时候会出现一些特殊的字符，比如带空格的路径等。另外，通过 URL 查询字符串传递数据时，也可能会出现特殊字符，例如，用 http://server/a.aspx?a=张三&b=12 传递数据时，在有些浏览器上就不能正确得到数据，这时就需要对字符串进行 URL 编码。





图 7-3　应用 HtmlEncode 方法的运行结果

例如：

```
string url;
url="http://myserver/1.aspx?a=";
url+=Server.UrlEncode("张 三");
url+="&b="+Server.UrlEncode ("ab   cd e");
Response.Write(url);
```

执行结果如图 7-4 所示。



图 7-4　应用 UrlEncode 方法的运行结果

（3）MapPath 方法

在页面中，一般使用的是虚拟路径，但是在连接 Access 数据库或其他文件操作时就必须使用物理路径。物理路径可以在程序中直接写出，但有时候不利于网站的移植，而利用 Server 对象的 MapPath 方法可以将虚拟路径转换为物理路径，既方便了网站的移植，又满足了程序的需要。

例如：

```
Response.Write("当前目录物理路径："+Server.MapPath("."));
Response.Write("<br>");
Response.Write("上级目录物理路径："+Server.MapPath("..."));
Response.Write("<br>");
Response.Write("网站根物理路径："+Server.MapPath("/"));
```

执行结果如图 7-5 所示。

Server 对象的常用属性和方法如表 7-5 所示。





图 7-5　应用 MapPath 方法的运行结果

表 7-5　Server 对象的常用属性和方法

名　　称	功 能 说 明
MachineName 属性	获取服务器的计算机名称
ScriptTimeout 属性	获取或设置文件最长执行时间(以秒计)
CreatObject 方法	创建 COM 对象的一个服务器实例
Execute 方法	使用另一个页执行当前请求
HtmlEncode 方法	对要在浏览器中显示的字符串进行编码
HtmlDecode 方法	对已被编码以消除无效 HTML 字符的字符串进行解码
UrlEncode 方法	对指定字符串以 URL 格式进行编码
UrlDecode 方法	对 URL 格式字符串进行解码
MapPath 方法	将虚拟路径转换为物理路径
Transfer 方法	终止当前页的执行，并开始执行新的请求页

7.1.8　Cookie 对象

Cookie 对象可用于保存客户端浏览器请求的服务器页面，也可以用它存放非敏感性的用户信息，信息的保存时间可以根据用户需要进行设置；但 Cookie 和其他对象的最大不同是 Cookie 将信息保存在客户端，而 Session 和 Application 是保存在服务器端。也就是说，无论何时用户连接到服务器，Web 站点都可以访问 Cookie 的信息。这样，既方便用户的使用，也方便了网站对用户的管理。如果没有设置 Cookie 的有效日期，那么它们仅保存到关闭浏览器程序为止；如果将 Cookie 对象的 Expires 属性设置为 MinValue，则表示 Cookie 永远不会过期。Cookie 存储的数据量受限制，大多数浏览器支持的最大容量为 4096B，因此，一般不用 Cookie 对象来保存数据集或其他大量数据。并非所有的浏览器都支持 Cookie，并且数据信息是以明文的形式保存在客户端计算机中，因此最好不要保存敏感的、未加密的数据，否则会影响网络的安全性。存储 Cookie 变量，可以通过 Response 对象的 Cookies 集合，需要注意的是，Cookie 对象不属于 Page 类，所以用法和其他对象不同。

语法如下：

```
HttpCookie coki1=new HttpCookie("abc");         //创建 Cookie 对象 coki1
```



```
coki1.Value="白菜";                          //为 coki1 对象的 Value 属性赋值白菜
Response.Cookies.Add(coki1);                  //把 coki1 对象加入 Cookies 集合中
```

要取回 Cookie，可以使用 Request 对象的 Cookies 集合，并将指定的 Cookies 集合返回。语法如下：

```
Response.Write(Request.Cookies["abc"].Value );  //获取 Cookie 对象的值并输出
```

1．Cookie 对象的常用属性

（1）Expires 属性

Expires 属性获取或设置 Cookie 的过期日期和时间。

例如，将 Cookie 的过期时间设置为当前时间之后 20 秒钟，代码如下：

```
HttpCookie cookie=new HttpCookie("username");      //声明一个 Cookie 变量
cookie.Value="白菜";                               //赋值给这个 cookie 变量
DateTime time=DateTime.Now;                         //获取当前时间
TimeSpan TSpan=new TimeSpan(0, 0, 0, 20);           //设置时间间隔
cookie.Expires=time.Add(TSpan);                     //cookie 的过期时间
```

（2）Name 属性

Name 属性获取或设置 Cookie 的名称。

（3）Value 属性

Value 属性获取或设置单个 Cookie 值。

（4）Path 属性

Path 属性获取或设置要与当前 Cookie 一起传输的虚拟路径。

例如，要获得与当前 Cookie 一起传输的虚拟路径可通过调用 Cookie 对象的 Path 属性来实现。要实现以上操作，可在创建的 Web 窗体的后台代码中输入代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie cookie=new HttpCookie("test");           //声明一个 Cookie 变量
    cookie.Value="cookieTest";                          //赋值给这个 cookie 变量
    Response.Cookies.Add(cookie);                       //添加 cookie
    Response.Write(Request.Cookies["test"].Path);       //页面输出
}
```

2．Cookie 对象的方法

（1）ToString 方法

ToString 方法返回当前 Object 的 String。

（2）Equals 方法

Equals 方法有两种重载形式：一种确定指定的 Object 是否等于当前的 Object；另外一种是确定指定的两个 Object 是否相等，如果相等，则为 true，否则为 false 。

例如，在创建的 Web 窗体的后台代码中输入如下代码，运行可以看到输出 Cookie 值相等。



```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie cookie1=new HttpCookie("test1");          //声明一个 cookie1 变量
    cookie1.Value="cookieTest";                          //赋值给这个 cookie1 变量
    Response.Cookies.Add(cookie1);                       //添加 cookie1
    HttpCookie cookie2=new HttpCookie("test2");          //声明一个 cookie2 变量
    cookie2.Value="cookieTest";                          //赋值给这个 cookie2 变量
    Response.Cookies.Add(cookie2);                       //添加 cookie2
    if(Equals(Request .Cookies["test1"].Value,Request.Cookies["test2"].Value))
        Response.Write ("Cookie 值相等");
    else
        Response.Write("Cookie 值不相等");
}
```

7.2　任务实施

7.2.1　登录页面的完善

本任务将对前面设计的登录页面 AdminLogin.aspx 进行完善，在前面设计的基础上加上验证码的验证，如图 7-6 所示。



图 7-6　AdminLogin.aspx 的界面

界面设计：在 AdminLogin.aspx 页面的源界面下添加如下代码。

```
<tr>
    <td width="290" height="49" ><div align="right">验证码：</div></td>
    <td width="205" height="49">
        <asp:TextBox ID="txtAdminCode" runat="server" ></asp:TextBox></td>
    <td width="205"> <asp:Label ID="labCode" runat="server"
    BackColor="#FFC0FF">8888</asp:Label></td>
</tr>
```

功能实现代码：

(1) 首先创建一个公共类 CommonClass.cs，实现随机码的产生，具体如下：

① 在解决方案资源管理器中右击，从弹出的快捷菜单中选择“新建文件夹”命令，将新建的文件夹命名为 App_code。



② 右击 App_code 文件夹，从弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，在其中选择“类”选项，在“名称”文本框中输入“CommonClass. cs”，如图 7-7 所示。



图 7-7 添加新项文件夹

③ 在 CommonClass. cs 中添加代码如下：

```
///<summary>
///实现随机验证码
///</summary>
///<param name="n">显示验证码的个数</param>
///<returns>返回生成的随机数</returns>
public string RandomNum(int n) //
{
    //定义一个包括数字、大写英文字母和小写英文字母的字符串
    string strchar="0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,
    Z,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z";
    //将 strchar 字符串转化为数组
    //String.Split 方法返回包含此实例中的子字符串(由指定 Char 数组的元素分隔)的 String
      数组
    string[] VcArray=strchar.Split(',');
    string VNum="";
    //记录上次随机数值,尽量避免产生几个一样的随机数
    int temp=-1;
    //采用一个简单的算法以保证生成随机数的不同
    Random rand=new Random();
    for (int i=1; i<n+1; i++)
    {
        if (temp!=-1)
        {
            //unchecked 关键字用于取消整型算术运算和转换的溢出检查
            //DateTime.Ticks 属性获取表示此实例的日期和时间的刻度数
            rand=new Random(i * temp * unchecked((int)DateTime.Now.Ticks));
        }
        //Random.Next 方法返回一个小于所指定最大值的非负随机数
        int t=rand.Next(61);
```






```
            if (temp!=-1 && temp==t)
            {
                return RandomNum(n);
            }
            temp=t;
            VNum+=VcArray[t];
        }
        return VNum;                //返回生成的随机数
    }
```

(2) 在 AdminLogin. aspx. cs 文件中添加功能代码(圈起来的是新加代码,其他为原有代码),如图 7-8 所示。

```
public partial class admin_AdminLogin : System.Web.UI.Page
{
    CommonClass CC = new CommonClass();
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            this.labCode.Text = CC.RandomNum(4);//产生验证码
        }
    }
    protected void btnAdminLogin_Click(object sender, EventArgs e)
    {
        if (txtAdminCode.Text.Trim() == labCode.Text.Trim())
        {
            if (AdminManager.AdminLogin(this.txtName.Text, this.txtPwd.Text))
            {
                Session["LoginUser"] = this.txtName.Text;

                string strRedirect;
                //表单验证所指定的路径
                strRedirect = Request["ReturnUrl"];
                //表单验证
                System.Web.Security.FormsAuthentication.SetAuthCookie(this.txtName.Text, true);
                if (strRedirect == null)
                    Response.Redirect("~/admin/ListAllStudents.aspx");
                Response.Redirect(strRedirect);
            }
            else
            {
                Response.Write("<script>alert('用户名或密码错误!');</script>");
            }
        }
        else
        {
            Response.Write("<script>alert('验证码输入有误,请重新输入!');</script>");
        }
    }
    protected void btnReset_Click(object sender, EventArgs e)
    {
```

图 7-8 在 AdminLogin. aspx. cs 文件中添加功能代码

7.2.2 网页浏览计数器

新建一个网站,在解决方案资源管理器中网站名上右击,从弹出的快捷菜单中选择“添加新项”命令,打开“添加新项”对话框,如图 7-9 所示。选择“全局应用程序类”选项,在“名称”文本框中输入“Global. asax”,然后单击“添加”按钮。在 Global. asax 文件中添加代码如下:

```
void Application_Start(object sender, EventArgs e)
{
    //在应用程序启动时运行的代码
    Application["coount"]=0;
```





图 7-9　添加 Global.asax 文件

```
        Application.Lock();
        Application["count"]=(int)Application["count"]+1;
        Application.UnLock();
    }
```

在 default.aspx 页面上添加一个 Label 控件，ID 属性设置为 Label1，Text 属性设置为空。在代码页中添加代码如下：

```
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text="您是第"+Application.Count+"位访客";
    }
```

运行结果如图 7-10 所示。



图 7-10　网页计数器运行结果

练　　习

1. 单项选择题

ASP.NET 代码 Response.Write(Server.htmlEncode("＜H1＞HtmlEncode 样例＜/H1＞"))输出结果为________。

A. 在窗口打印“HtmlEncode 样例”



B. 在窗口打印“<H1>HtmlEncode 样例</H1>”
C. 在窗口打印“H1HtmlEncode 样例 H1”
D. 出现错误信息，说明嵌入的串中包含非法字符

2. 简答题

(1) 简要说明 global. asax 文件中下面几个函数方法在什么情况下启动。

```
void Session_OnStart()
void Session_OnEnd()
public void Application_OnStart()
public void Application_OnEnd()
```

(2) 简要阐述 Response 对象、Request 对象、Session 对象和 Application 对象的功能。
(3) 列举 ASP. NET 页面之间传递值的几种方式。

实　　训

实训目的

(1) 熟练掌握 global. asax 文件的使用。
(2) 掌握 Session 和 Application 对象的使用。
(3) 统计网站的当前在线人数。

实训内容

(1) 创建网站并添加 global. asax 文件。
(2) 添加页面显示当前网站在线人数。

实训指导

(1) 创建网站并添加 global. asax 文件，具体方法与前面网页浏览计数器的创建过程一样。

(2) 在 global. asax 文件中添加代码如下：

```
void Application_Start(object sender, EventArgs e)
{
    //在应用程序启动时运行的代码
    Application.Lock();
    Application["online"]=0;
    //Application["online"]=0;
    Application.UnLock();
}
void Session_Start(object sender, EventArgs e)
{
    //在新会话启动时运行的代码
    Application.Lock();
    Application["online"]=(int)Application["online"]+1;
```



```
    Application.UnLock();
}
void Session_End(object sender, EventArgs e)
{
    //在会话结束时运行的代码
    //注意：只有在 Web.config 文件中的 sessionstate 模式设置为
    //InProc 时,才会引发 Session_End 事件。如果会话模式设置为 StateServer
    //或 SQLServer,则不会引发该事件
    Application.Lock();
    Application["online"]= (int)Application["online"] -1;
    Application.UnLock();
}
```

(3) 在 default.aspx 页面上添加一个 Label 控件，设置 Text 属性为空。

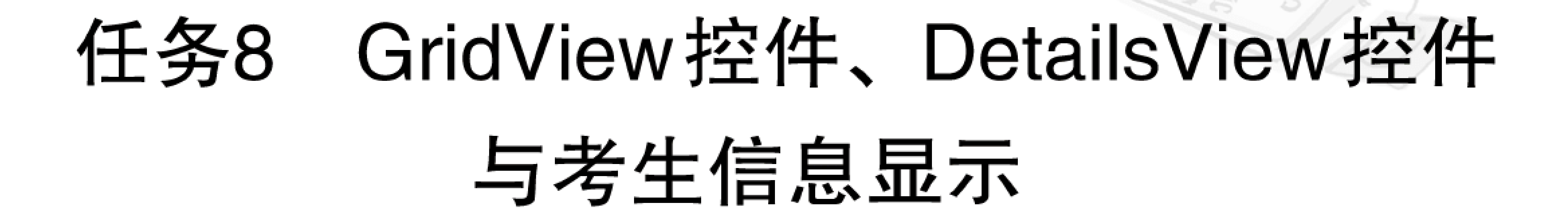
(4) 在 default.aspx.cs 页中添加代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text="当前在线"+Application["online"].ToString ()+"人";
}
```

(5) 页面运行效果如图 7-11 所示。



图 7-11 统计网站在线人数的程序运行结果



任务8　GridView控件、DetailsView控件与考生信息显示

技能目标

能使用 GridView 控件为在线考试系统制作考生信息显示列表，能使用 DetailsView 控件为在线考试系统制作显示考生详细信息的页面。

知识目标

掌握 GridView 控件的用法，掌握 DetailsView 控件的用法；会使用 GridView 控件和 DetailsView 控件显示考生信息。

任务描述

从数据库读取信息并显示在页面上是 Web 开发中很重要的一个部分，要从数据库读取并显示各种信息必须掌握 ASP. NET 提供的数据源控件和数据绑定控件的功能和用法，并根据具体的情况使用不同的数据源控件和数据绑定控件实现信息的显示。

本章任务如下：

- 使用 GridView 控件以列表形式显示考生信息；
- 使用 DetailsView 控件显示考生的详细信息。

8.1　知 识 准 备

8.1.1　数据源控件

数据源控件是 ASP. NET 2.0 中引入的一种新型服务器控件，它们是数据绑定体系结构的一个关键部分，能够通过数据绑定控件来提供声明性编程模型和自动数据绑定行为。在之前的 ASP. NET 版本中存取数据库需要使用 ADO. NET 对象，用程序代码连接、打开、操作和显示数据表的记录数据；ASP. NET 2.0 则提供了全新的数据源控件，可以将程序代码封装成服务器端控件，只需在控件中声明数据源，就可以存取数据表的记录数据。数据源控件封装所有获取和处理数据的功能，主要包括连接数据源，使用 Select、Update、Delete 和 Insert 等对数据进行管理。

ASP. NET 2.0 提供了多种数据源控件，最常用的有以下 5 种。



- SqlDataSource：存取关系型数据库的数据源。可以是 SQL Server、Access 和 Oracle 等，如果使用 SQL Server，控件自动使用 SqlClient 类来最佳化存取数据库。
- AccessDataSource：存取 Microsoft Access 数据库。
- ObjectDataSource：存取业务对象的数据源，可以在多层结构中存取中间层的数据源。
- XmlDataSource：存取 XML 文件的数据源。
- SiteMapDataSource：建立网站地图的只读数据源。

数据源控件在 Visual Studio 的工具箱窗口中的位置如图 8-1 所示，可以在设计视图下通过拖动在页面中添加这些控件。



图 8-1　数据源控件

8.1.2　数据绑定控件

数据源控件可以获取数据，但是它们不能单独使用。要想使用数据源控件，必须将它们绑定到数据绑定控件。数据绑定控件将数据以标记的形式呈现给请求数据的浏览器。数据绑定控件可以绑定到数据源控件，并自动在页请求生命周期的适当时间获取数据。数据绑定控件可以利用数据源控件提供的功能，包括排序、分页、缓存、筛选、更新、删除和插入。数据绑定控件通过其 DataSourceID 属性指定要绑定到的数据源控件。

数据绑定控件的层次结构如图 8-2 所示。

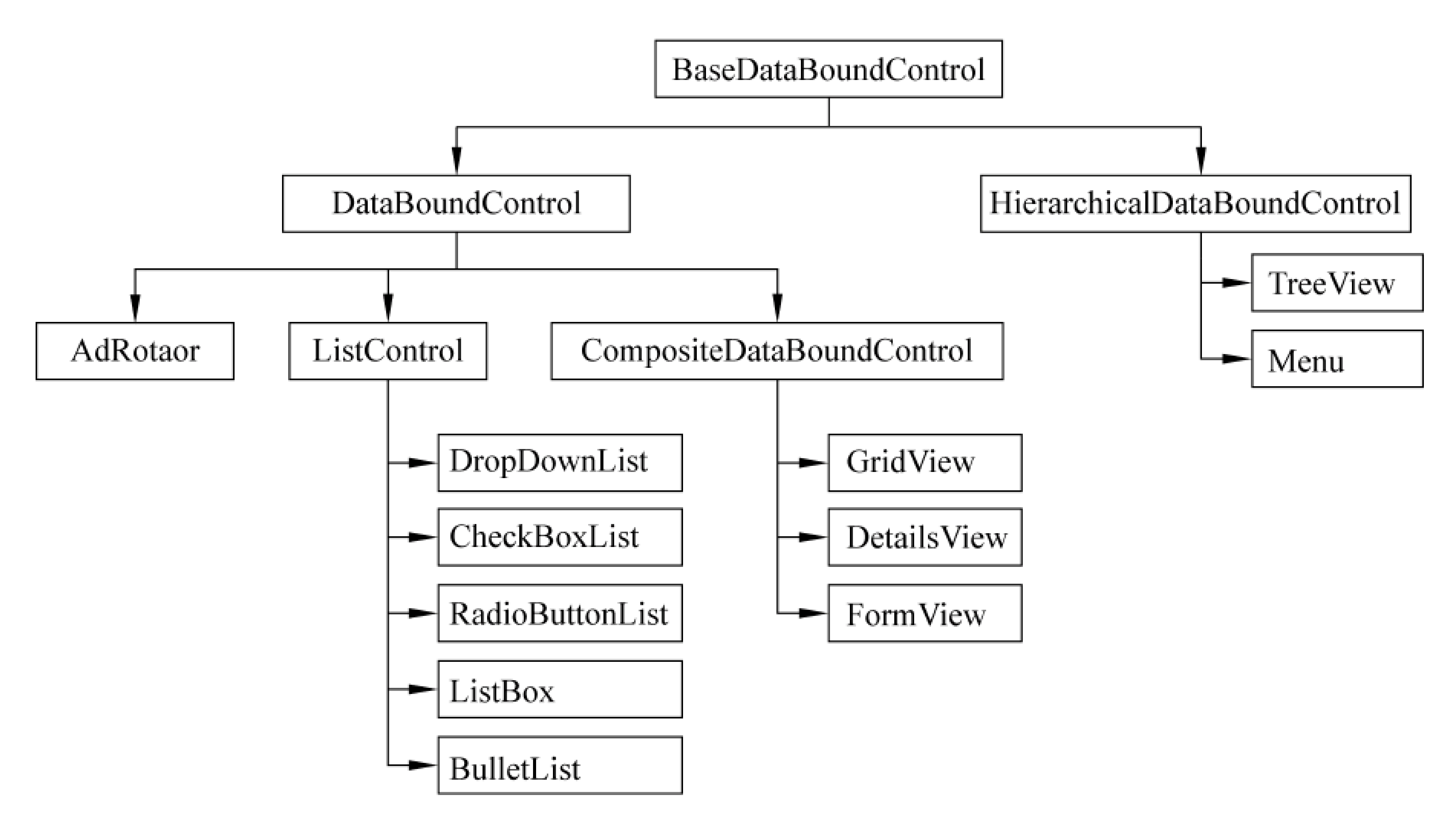


图 8-2　数据绑定控件的层次结构

常用的数据绑定控件包括以下几种。

- 列表：以各种列表形式呈现数据，如 DropDownList、CheckBox、RadioButtonList、ListBox 等。
- GridView：以表的形式呈现数据，支持在不编写代码的情况下对数据进行编辑、更新、删除、排序等操作。



- DetailsView：以表格布局的形式一次显示一条记录的详细信息，并允许进行编辑、删除和插入等操作。
- DataList：以表的形式呈现数据。通过项模板来控制显示的外观。
- 导航：通过绑定站点地图或者XML文件来显示站点的导航信息，包括TreeView和Menu。

在本章的任务中要用到GridView控件和DetailsView控件，下面对这两种控件进行介绍。

1．GridView控件

在动态网页开发中，以表格形式显示数据是一项很常见的任务。ASP.NET提供了许多工具用来显示表格形式的数据，GridView控件是其中最重要的工具之一。通过使用GridView控件，并绑定相关的数据源控件，可以对各种数据源（如数据库、XML文件和公开数据的业务对象等）进行显示、编辑和删除等多种不同的操作。

使用GridView可以完成以下操作。

1）通过数据源控件自动绑定和显示数据。

2）通过数据源控件对数据进行选择、排序、分页、编辑和删除。

（1）GridView的属性

GridView的常用属性如下。

- AllowPaging：获取或设置一个值，该值指示是否启用分页功能。
- AllowSorting：获取或设置一个值，该值指示是否启用排序功能。
- Columns：获取表示GridView控件中列字段的DataControlField对象的集合。
- Controls：获取复合数据绑定控件内的子控件的集合。
- DataKeyNames：获取或设置一个数组，该数组包含了显示在GridView控件中的项的主键字段的名称。
- DataKeys：获取一个DataKey对象集合，这些对象表示GridView控件中的每一行的数据键值。
- DataMember：当数据源包含多个不同的数据项列表时，获取或设置数据绑定控件绑定到的数据列表的名称。
- DataSource：获取或设置对象，数据绑定控件从该对象中检索其数据项列表。
- DataSourceID：获取或设置控件的ID，数据绑定控件从该控件中检索其数据项列表。
- EditIndex：获取或设置要编辑的行的索引。
- GridLines：获取或设置GridView控件的网格线样式。
- Page：获取对包含服务器控件的Page实例的引用。
- PageCount：获取在GridView控件中显示数据源记录所需的页数。
- PageIndex：获取或设置当前显示页的索引。
- PageSize：获取或设置GridView控件在每页上所显示的记录的数目。
- Rows：获取表示GridView控件中数据行的GridViewRow对象的集合。



(2) GridView 的事件

GridView 的常用事件如下。

- RowCommand：在 GridView 控件中单击某个按钮时发生。此事件通常用于在该控件中单击某个按钮时执行某项任务。
- PageIndexChanging：在单击页导航按钮时发生，但在 GridView 控件执行分页操作之前。此事件通常用于取消分页操作。
- PageIndexChanged：在单击页导航按钮时发生，但在 GridView 控件执行分页操作之后。此事件通常用于在用户定位到该控件中不同的页之后需要执行某项任务时。
- SelectedIndexChanging：在单击 GridView 控件内某一行的 Select 按钮（其 CommandName 属性设置为"Select"的按钮）时发生，但在 GridView 控件执行选择操作之前。此事件通常用于取消选择操作。
- SelectedIndexChanged：在单击 GridView 控件内某一行的 Select 按钮时发生，但在 GridView 控件执行选择操作之后。此事件通常用于在选择了该控件中的某行后执行某项任务。
- Sorting：在单击某个用于对列进行排序的超链接时发生，但在 GridView 控件执行排序操作之前。此事件通常用于取消排序操作或执行自定义的排序例程。
- Sorted：在单击某个用于对列进行排序的超链接时发生，但在 GridView 控件执行排序操作之后。此事件通常用于在用户单击对列进行排序的超链接之后执行某项任务。
- RowDataBound：在 GridView 控件中的某个行被绑定到一个数据记录时发生。此事件通常用于在某个行被绑定到数据时修改该行的内容。
- RowCreated：在 GridView 控件中创建新行时发生。此事件通常用于在创建某个行时修改该行的布局或外观。
- RowDeleting：在单击 GridView 控件内某一行的 Delete 按钮（其 CommandName 属性设置为"Delete"的按钮）时发生，但在 GridView 控件从数据源删除记录之前。此事件通常用于取消删除操作。
- RowDeleted：在单击 GridView 控件内某一行的 Delete 按钮时发生，但在 GridView 控件从数据源删除记录之后。此事件通常用于检查删除操作的结果。
- RowEditing：在单击 GridView 控件内某一行的 Edit 按钮（其 CommandName 属性设置为"Edit"的按钮）时发生，但在 GridView 控件进入编辑模式之前。此事件通常用于取消编辑操作。
- RowCancelingEdit：在单击 GridView 控件内某一行的 Cancel 按钮（其 CommandName 属性设置为"Cancel"的按钮）时发生，但在 GridView 控件退出编辑模式之前。此事件通常用于停止取消操作。
- RowUpdating：在单击 GridView 控件内某一行的 Update 按钮（其 CommandName 属性设置为"Update"的按钮）时发生，但在 GridView 控件更新记录之前。此事件通常用于取消更新操作。
- RowUpdated：在单击 GridView 控件内某一行的 Update 按钮时发生，但在 GridView 控件更新记录之后。此事件通常用来检查更新操作的结果。
- DataBound：此事件继承自 BaseDataBoundControl 控件，在 GridView 控件完成到

数据源的绑定后发生。

(3) GridView控件的字段类型

GridView是由一组字段(Field)组成的,它们都指定了来自DataSource中的什么属性需要用到自己的输出呈现中。GridView共支持七种字段类型,这七种字段类型都具有五大类属性,如表8-1所示。

表8-1　字段类型的五大类属性

属　性	说　明
可访问性(Accessibility)	设置AccessibleHeaderText
外观(Appearance)	设置如HeaderText、FooterText、HeaderImageUrl
行为(Behavior)	设置行为属性,其中大多为布尔值true/false
数据(Data)	设置数据源字段与字符串格式化
样式(Style)	包括ControlStyle、HeaderStyle、FooterStyle、ItemStyle样式,分别用于设置控件、项、页眉及页脚的颜色、字体、宽度、对齐方式等样式的属性

下面将逐一介绍这七种字段的用法。

① BoundField

BoundField即数据绑定字段,BoundField字段的图标为 BoundField,它的主要作用是将DataSource数据源的字段数据以文本方式显示。BoundField对象在不同的数据绑定控件中会有不同的显示方式,如BoundField对象在GridView控件中显示为数据列,而在DetailsView控件中会显示为数据行。

在上述五大类属性中,BoundField字段常用的属性如下。

✍ DataField:指定对应的DataSource数据字段。

✍ DataFormatString:显示文本的字符串格式化,如显示成货币、科学符号。

✍ SortExpression:设置字段的排序键值。

✍ HtmlEncode:是否进行HtmlEncode编码。

其中,使用DataFormatString可以实现对绑定字段的格式的设置。常用的标准数值格式化字符串包括以下几种。

✍ {0:C}:显示货币符号格式。

✍ {0:D}:显示十进制数格式(限用于整数)。

✍ {0:yy-mm-dd}或{yy-MM-dd}或{yyyy-mm-dd}:显示日期,如2009-10-01。

小贴士

如果要使设定的格式起作用,必须使HtmlEncode属性的值为false,否则格式化无效。

② CheckBoxField

CheckBoxField即复选框字段,该字段的图标为 CheckBoxField 。它显示为CheckBox类型,通常用于布尔值true/false的显示。若数据源为true,则为勾选状态;若为false则未勾选。在显示状态下,默认CheckBoxField是被Disabled,故呈现的是灰色的只读状态,只有在编辑模式时才会变成Enabled,才能进行修改。

在上述五大类属性中,CheckBoxField字段常用的属性如下。



✍ DataField：指定对应的 DataSource 数据字段，即绑定至数据源的字段名称。

✍ Text：可以设置 CheckBox 右侧的说明文字。

✍ ReadOnly：在编辑模式时，设置 ReadOnly 属性可以防止被编辑。

📖小贴士

只有在数据源中存在布尔类型的数据时，才能使用该字段。

③ HyperLinkField

HyperLinkField 即超链接字段，该字段的图标为 HyperLinkField，用于将 DataSource 数据源字段数据显示成 HyperLink 超级链接，并可指定另外的 NavigateUrl 超链接，以导向实际的 URL 网址。

在上述五大类属性中，HyperLinkField 字段常用的属性如下。

✍ DataTextField：绑定数据源字段显示成超链接文字，可当作参数传递到 DataTextFormatString 属性之中格式化。

✍ DataTextFormatString：DataText 字段字符串的格式化。

✍ DataNavigateUrlFields：将数据字段绑定到超链接字段的 URL 属性，可当作参数传递到 DataNavigateUrlFormatString 属性中格式化。

✍ DataNavigateUrlFormatString：DataText 字段字符串的格式化。

✍ Text：超链接字段显示的文字，若 DataTextField 属性没有设置，会以 Text 文字显示。

④ ImageField

ImageField 即图像字段，该字段的图标为 ImageField，用于在数据绑定控件中显示图像字段。通常该字段绑定的内容是图片的路径，即指定图片的 URL 网址给 ImageField 的 DataImageUrlField 属性。

在上述五大类属性中，ImageField 字段常用的属性如下。

✍ DataAlternateTextField：设置绑定至 ImageField 对象 AlternateText 属性的数据源字段名称。

✍ DataAlternateTextFormatString：将 DataAlternateTextField 的字符串格式化。

✍ DataImageUrlField：设置绑定至 ImageField 对象 ImageUrl 属性的数据源字段名称。

✍ DataImageUrlFormatString：将 DataImageUrlField 的 URL 格式化。

✍ NullImageUrl：当 DataImageUrlField 属性值为 null 时，则以 NullImageUrl 属性的默认图片来替代。

⑤ ButtonField

ButtonField 即按钮字段，该字段的图标为 ButtonField，用于在数据绑定控件中显示命令按钮。根据控件的不同，它可以显示具有自定义按钮控件（如添加、删除、更新按钮等）的数据行或数据列，按下时会引发 RowCommand 事件。

在上述五大类属性中，ButtonField 字段常用的属性如下。

✍ ButtonType：按钮显示的形式，如 Button、Image、Link。

✍ DataTextField：将数据源字段数据绑定到 Button 按钮的文本属性中。



✍ DataTextFormatString：将 DataTextField 数据源字段值加以格式化。

✍ ImageUrl：当按钮形式为 Image 时，指定 Image 所在的 URL。

✍ CauseValidation：单击按钮时是否会引发 Validation 控件验证。

✍ CommandName：单击 ButtonField 按钮时所要运行的命令名称。

✍ ValidationGroup：ButtonField 按钮所要引发的 Validation Group 名称。

⑥ CommandField

CommandField 即命令字段，用来显示含有命令的 Button 按钮，包括了 Select、Edit、Update、Delete 命令按钮(DetailsView 的 CommandField 才支持 Insert 命令)。在外观上，CommandField 与 ButtonField 很相似，两者之间最大的差异在于 ButtonField 只是单纯地显示 Button 按钮而不具备内置的命令，所以 ButtonField 必须自行撰写相关程序，而 CommandField 可以利用内置的命令字段实现相应的命令操作。

在 GridView 中，CommandField 命令按钮字段支持“编辑、更新、取消”、“选取”与“删除”三种命令按钮，这三种命令按钮的图标分别是 编辑、更新、取消 、 选择 和 删除 。

小贴士

要使命令字段可以执行相应的命令操作，必须在相应的数据源控件中设定相应的 Select、Update、Insert 和 Delete 方法。

在上述五大类属性中，CommandField 字段常用的属性如下。

✍ ButtonType：按钮显示的形式，如 Button、Image、Link。

✍ ShowDeleteButton、ShowEditButton、ShowInsertButton 与 ShowSelectButton：显示及隐藏相应的命令按钮(true 或 false)。

✍ SelectText、InsertText、UpdateText、DeleteText、CancelText、EditText、NewText：设置不同命令按钮的文字标题。

✍ CancelImageUrl、 DeleteImageUrl、 EditImageUrl、 InsertText、 NewImageUrl、SelectImageUrl、UpdateImageUrl：当把 ButtonType 属性设为 ButtonType.Image 时，用于设置按钮的图像 URL 属性。

⑦ TemplateField

TemplateField 即模板字段，该字段的图标为 TemplateField ，用来显示用户自定义的模板内容。当 GridView 所预定的几种字段都无法满足需求时，就可以使用 TemplateField，而且在 TemplateField 中放入各种需要的控件。

模板字段的添加方式有两种：一种是像其他字段一样直接单击“添加”按钮添加；另一种方式是将预定义字段直接转换为模板字段，如图 8-3 所示。

TemplateField 是一种很重要的字段类型，它又包含了五种模板，分别如下。

✍ ItemTemplate：字段项目模板。

✍ AlternatingItemTemplate：字段交替项目模板。若设置这个字段后，奇数行会显示 ItemTemplate，偶数行显示 AlternatingItemTemplate。

✍ EditItemTemplate：编辑项目模板。

✍ HeaderTemplate：表头模板。

✍ FooterTemplate：表尾模板。





图 8-3　添加模板列

模板列建立好之后，可以通过在 GridView 上右击，从弹出的快捷菜单中选择“编辑模板”命令来编辑模板。以下是模板列未编辑之前的通用代码。

```
<asp:TemplateField>
       <ItemTemplate>
             <asp:控件>...</asp>
       </ItemTemplate>
       <EditItemTemplate>
             <asp:控件>...</asp>
       </EditItemTemplate>
       <HeaderTemplate>
             <asp:控件>...</asp>
       </HeaderTemplate>
       <AlternatingItemTemplate>
             <asp:控件>...</asp>
       </AlternatingItemTemplate>
       <FooterTemplate>
             <asp:控件>...</asp>
     </FooterTemplate>
</asp:TemplateField>
```

编辑的时候可以在相应的模板项中添加控件。在模板项中添加了控件之后的效果如图 8-4 所示。

（4）Eval 方法和 Bind 方法

ASP. NET 2. 0 的 Data-Binding 数据绑定语法表示符为“<％ ＃％ >”，里面通常会搭配 Eval 或 Bind 指令，如<％＃Eval("ProductID")％>或<％＃Bind("ProductID")％>，其中 ProductID 则为从数据库中获取到的数据源字段。

Eval 和 Bind 的区别如下。

✍ Eval：用于单向数据绑定，数据是只读显示。因此如果只是显示数据可以采用此种方法，通常 BoundField

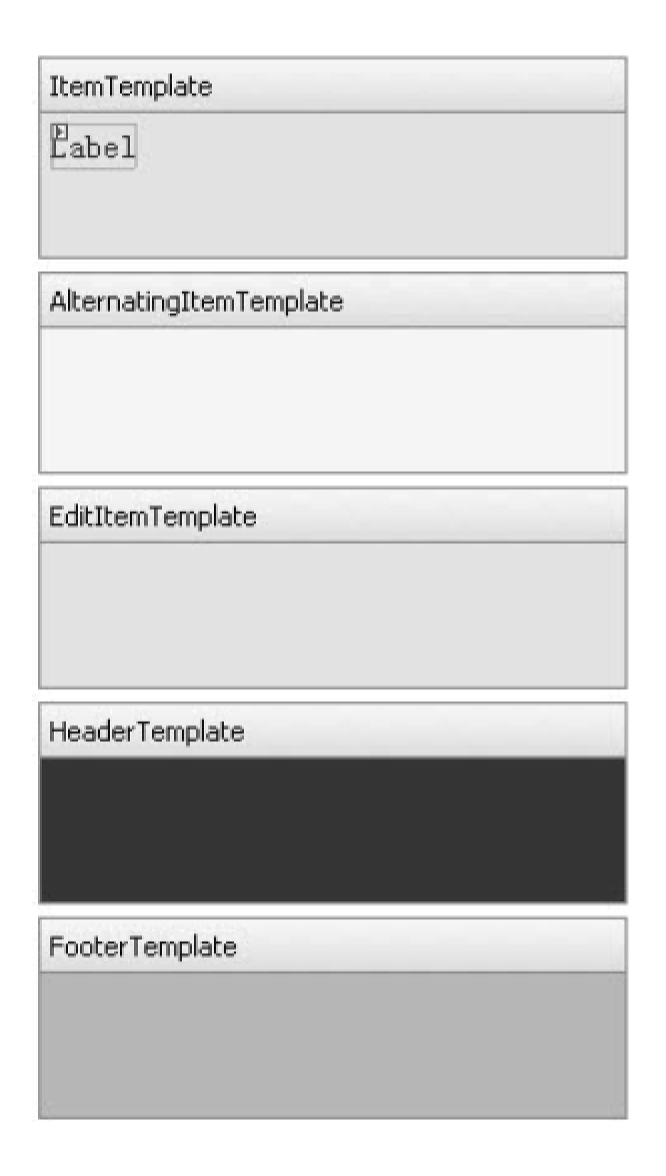


图 8-4　编辑模板列



字段默认是单向数据绑定。

- Bind：用于双向数据绑定，不但能读取数据，更具有 Insert、Update、Delete 功能，所以如果需要编辑更新、添加与删除功能必须使用本方法。

2. DetailsView 控件

DetailsView 控件一次呈现一条表格形式的记录，并提供浏览多条记录以及插入、更新和删除记录的功能。DetailsView 控件通常用于更新和插入新记录，并且通常在主/详细方案中使用，DetailsView 控件通常用来显示许多记录列表的详细信息。比如在任务 8 中显示了考生的所有信息，那么就可以利用 DetailsView 控件显示每条记录的详细信息。另外，即使 DetailsView 控件的数据源公开了多条记录，该控件一次也仅显示一条数据记录。

DetailsView 控件在很多方面与 GridView 控件非常类似，包括属性、事件、字段类型、模板列以及绑定方法等。

下面介绍 DetailsView 控件常用的属性和事件。DetailsView 控件常用的属性和事件主要包括以下几种。

- 属性 DefaultMode：设置或获取控件默认的状态（模式）。该属性为枚举值，分为 ReadOnly（显示）、Edit（修改）和 Insert（添加）。
- 属性 DataKey：数据的主键。
- 属性 DataKeyNames：设置或获取一个字符串，该字符串包括数据源中间的组合内容。
- 事件 ItemInserting：单击“添加”按钮，执行添加方法之前执行。
- 事件 ItemInserted：单击“添加”按钮，执行添加方法之后执行。
- 事件 ItemUpdating：单击“更新”按钮，执行更新方法之前执行。
- 事件 ItemUpdated：单击“更新”按钮，执行更新方法之后执行。

8.1.3　数据绑定方法

数据源控件在从数据库获取数据源之后，与数据绑定控件绑定的方法主要有两种。

（1）利用编码方式实现数据源的绑定

例如：

```
this.gridview1.DataSource=UserManager.GetAllusers();
this. gridview1.DataBind();
```

（2）指定数据源控件的 ID

在数据绑定控件的智能标记中选择已经配置好的数据源即可。

8.2　任务实施

8.2.1　在后台管理中显示考生信息

显示考生信息是考生管理中很重要的一项内容。利用 GridView 控件可以显示考生信息，考生信息显示的效果如图 8-5 所示。





图 8-5　考生列表信息

此模块的功能要求如下：

（1）分页显示考生信息列表（学号、考生姓名、班级等），每页显示 10 条记录；

（2）单击“详细”链接指向考生信息的详细页面；

（3）单击“删除”按钮可以实现单条记录的删除。

该模块可以利用 GridView 控件绑定 ObjectDataSource 数据源控件来实现，下面将详细介绍实现的具体步骤。

1. 创建考生信息列表页面

打开在线考试系统，右击目录 admin，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，如图 8-6 所示。设置窗体名称为“ListAllStudents”，选中“选择母版页”复选框。

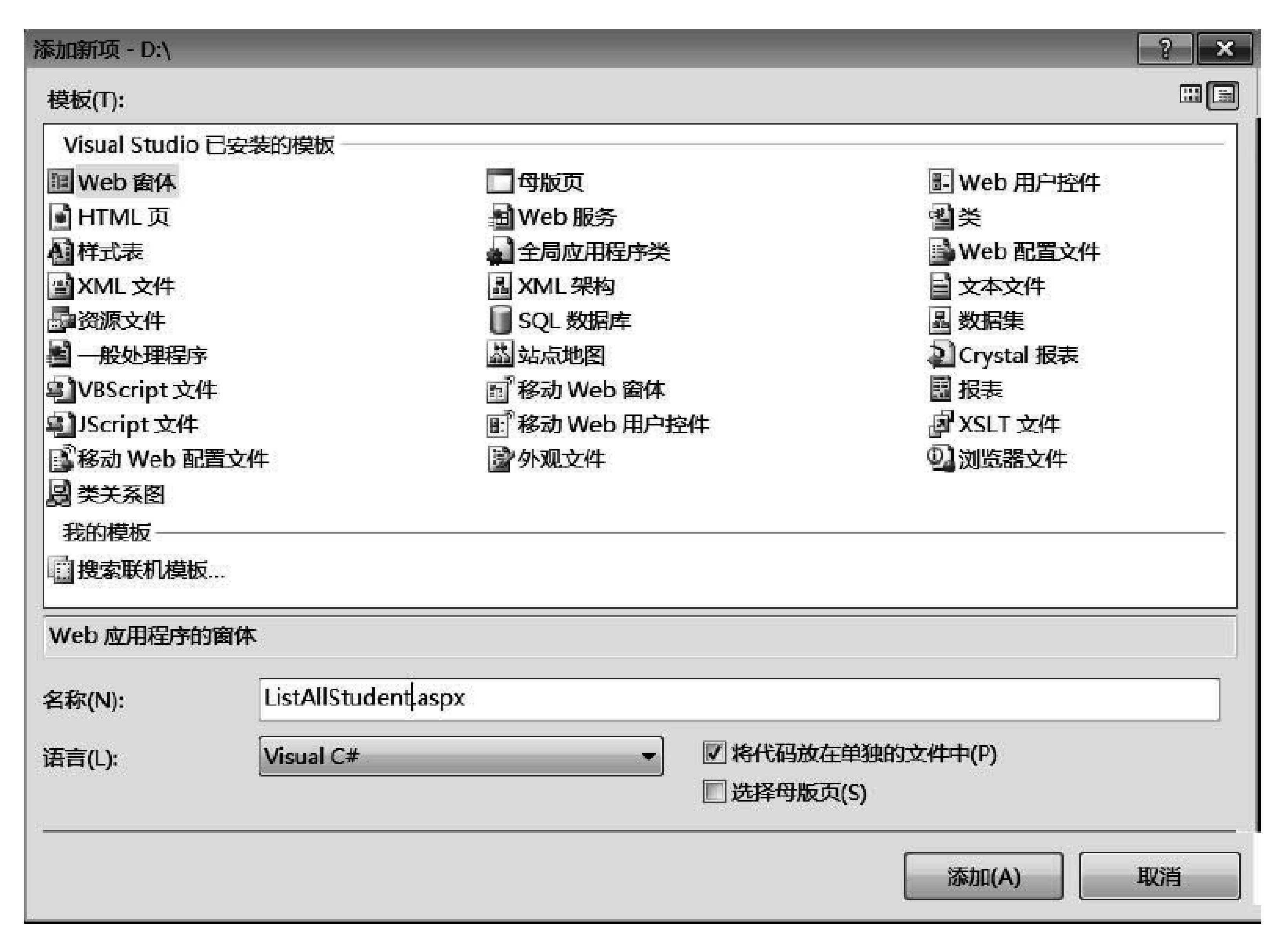


图 8-6　创建 ListAllStudent 窗体



在设计界面向窗体中拖入 GridView 控件，选择该控件，修改其 ID 属性值为“gvStudent”。

设置 GridView 的外观。为了使显示美观，可以选择“自动套用格式”命令，并在智能标记中选择一种自动套用的格式，打开“自动套用格式”对话框，如图 8-7 所示，在其中选择一种喜欢的样式。这里选择“石板”样式。



图 8-7 设置自动套用格式

2. 考生信息列表的绑定

为了使用 GridView 显示考生信息列表，需要将 GridView 绑定到相应的数据源，才能最终显示。实现的具体步骤如下：

(1) 将 ObjectDataSource 控件拖入 ListAllStudents. aspx 页面，并修改其 ID 值为“odsStudents”，然后在智能标记中选择“配置数据源”命令，打开如图 8-8 所示的“配置数据源”对话框，并选择业务逻辑层的 StudentManager 类。



图 8-8 选择业务对象

(2) 单击“下一步”按钮，进入“定义数据方法”对话框，可以设置需要的增、删、改、查方法。



由于我们需要显示考生的信息列表以及删除某条考生信息，所以在SELECT选项卡中选择“GetAllStudent()，返回IList＜examStudent＞”选项用于返回所有考生信息，在DELETE选项卡中选择DeleteExamStudentById(String Stu_id)即可按照考生ID来删除考生的记录，如图8-9所示。

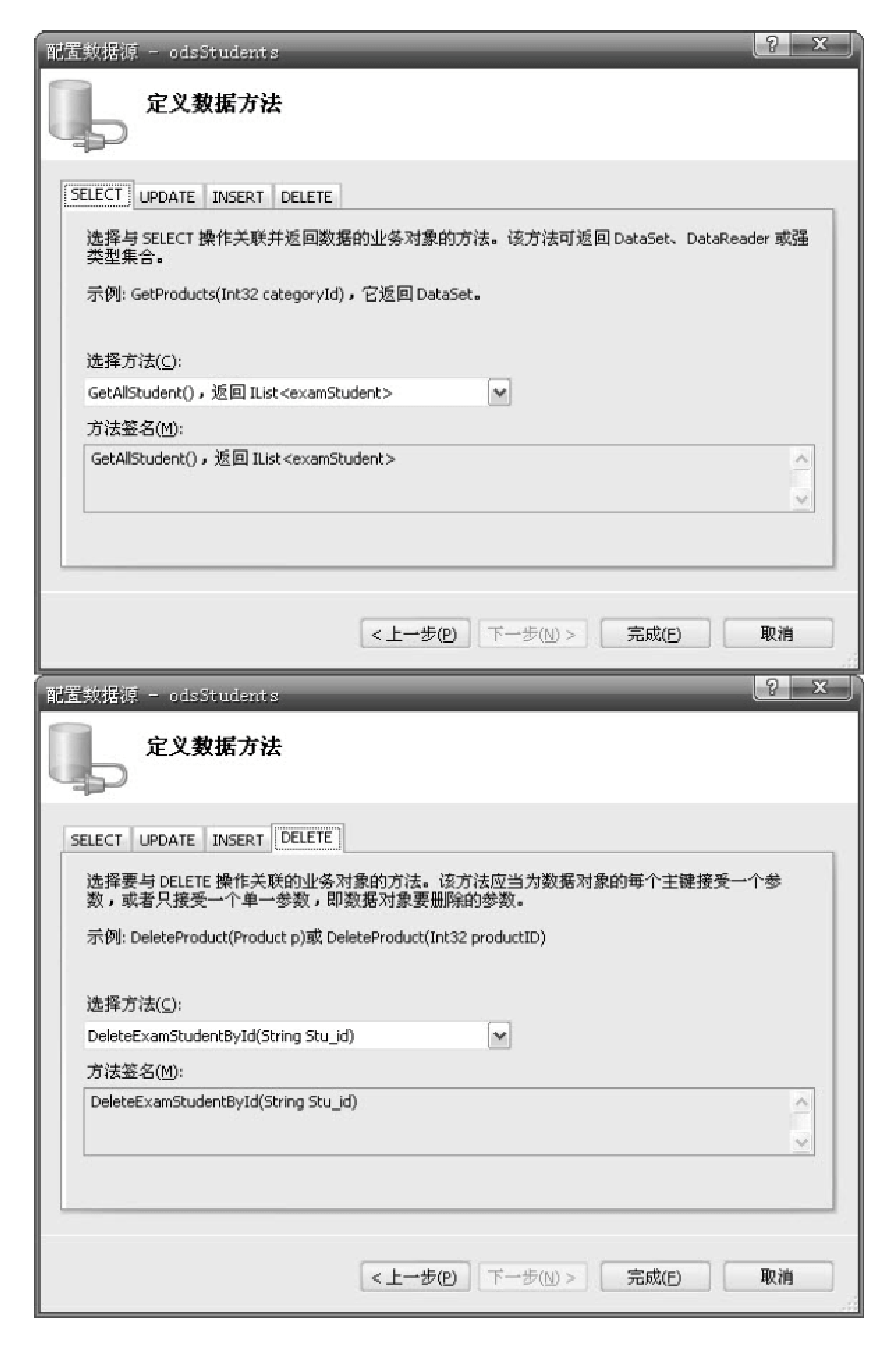


图8-9　定义数据方法

(3) 单击“完成”按钮，数据源即可配置成功。

(4) 选择GridView控件，然后在智能标记中的“选择数据源”下拉列表框中选择odsStudents数据源，如图8-10所示，之后即可实现所有考生信息的显示。

(5) 启用分页和删除功能。因为考生信息很多，因此要分页显示，在智能标记中选中“启用分页”复选框即可；另外还要按照考生学号来删除考生记录，因此在智能标记中选中“启用删除”复选框。效果如图8-11所示。

📖小贴士

如果要使“启用删除”复选框可用，在ObjectDataSource数据源中一定要设置删除的方法，否则“启用删除”不可用。



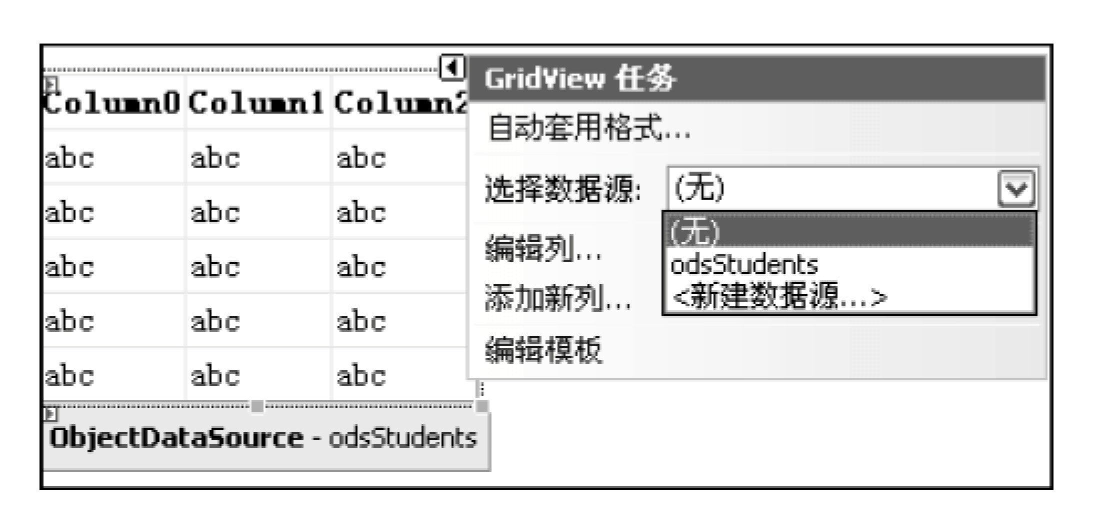


图 8-10　选择数据源



图 8-11　启用分页和删除功能

3. 编辑字段

（1）刚才显示的是所有的字段，为了显示需要显示的字段，可以在智能标记中选择“编辑列”命令，如图 8-12 所示。

在打开的“字段”对话框中，可以设置显示的内容、标题以及显示顺序，如图 8-13 所示。在此对话框中，可以通过上下箭头来调整字段显示的顺序，按钮的作用是用于删除字段。在右侧的 BoundField 属性选项区域中可设置相应的属性，包括修改显示的标题头，调整显示字段之间的大小等操作。



图 8-12　编辑列



图 8-13　“字段”对话框

（2）设置“班级”字段。由于“班级”字段是一个外键字段，直接地绑定列无法正常显示班级信息，因此需要先将绑定列转换为模板列，然后设置 ItemTemplate 中的 Label 的 DataBinding 为 Eval("examStuClass.Class_name")，如图 8-14 所示。

（3）设置“详细”字段。首先添加一个 HyperLinkField 字段，然后将 HeaderText 设置为“详细”，如图 8-15 所示。将 DataNavigateUrlField 设置为 Stu_Id，DataNavigateUrlFormatString 设置为～/admin/StudentDetails.aspx?sId={0}。





图 8-14 “班级”字段的绑定设置

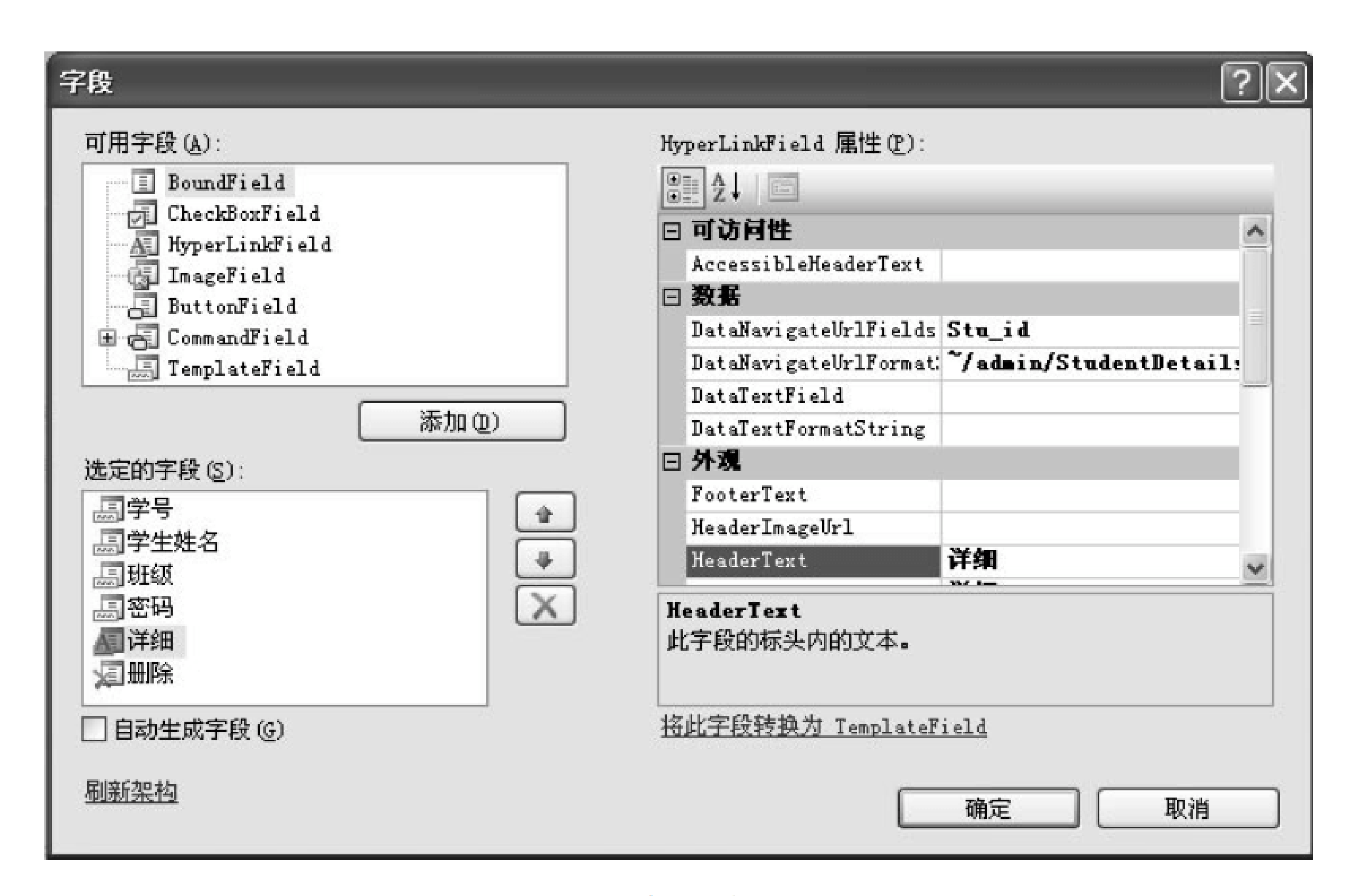


图 8-15 “详细”字段的设置

8.2.2 考生详细信息显示

在 ListAllStudents.aspx 页面中，我们利用 GridView 控件显示了所有考生的信息；除了要显示所有考生的基本信息之外，在一些情况下，还要显示考生的详细信息，编辑考生的信息。此时，可以利用 DetailsView 控件来实现。下面将详细说明利用 DetailsView 控件来实现显示单个考生信息并进行编辑的页面 StudentDetails.aspx。

显示考生详细信息页是考生信息管理中很重要的一项内容，利用 DetailsView 控件可以显示考生的详细信息并且可以利用内置功能对信息进行编辑。考生详细信息显示的效果如图 8-16 所示。

此模块的功能需求如下：

(1) 根据 ListAllStudents.aspx 页面传递过来的 ID 显示相应考生的详细信息；





图 8-16　考生详细信息页面

（2）能够对考生的详细信息进行编辑。

该模块可以利用 DetailsView 控件绑定 ObjectDataSource 数据源控件来实现，下面将详细介绍实现的具体步骤。

1. 创建考生详细信息列表页面

打开在线考试系统，右击目录 admin，从弹出的快捷菜单中选择“添加新项”命令，新建 Web 窗体，窗体名称为“StudentsDetails”，选中“选择母版页”复选框。

在设计界面向窗体中拖入 GridView 控件，选择该控件，修改其 ID 属性值为“gvStudent”。

2. 考生详细信息的绑定

（1）将 ObjectDataSource 控件拖入 StudentDetails. aspx 页面，并修改其 ID 值为：odsStudentDetail，然后在智能标记中选择“配置数据源”命令，打开如图 8-17 所示的“配置数据源”对话框，并选择业务逻辑层的 StudentManager 类。

（2）单击“下一步”按钮，进入“定义数据方法”对话框，可以设置需要的增、删、改、查方法，如图 8-18 所示。由于这里要显示考生的详细信息以及编辑该条考生信息，并且该信息是根据 ListAllStudents. aspx 页面传递过来的 ID 值来读取相应考生信息的，所以在 SELECT 选项卡中选择 GetStudentById(String Stu_id)方法用于显示某考生信息，在 UPDATE 选项卡中选择 ModifyStudentBasicInfo(String Stu_name，Int32 Class_id，String Stu_pwd，Int32 IsLogin，Int32 IsSubmit，String Stu_id)方法即可按照考生 ID 来修改考生的信息。

（3）单击“下一步”按钮，由于 ID 值是根据 ListAllStudents. aspx 页面传递过来的，因此需要在“定义参数”对话框中，指定“参数源”为 QueryString，QueryStringField 为 sId，如





图 8-17　选择业务对象



图 8-18　定义数据方法

图 8-19 所示。

(4) 单击“完成”按钮，获得某条考生详细信息的数据源即可配置成功。

(5) 将 DetailsView 控件拖入该页面，然后在智能标记中的“选择数据源”下拉列表框中选择 odsStudentDetail 数据源，如图 8-20 所示，之后即可实现单个考生信息的显示。绑定之后的效果如图 8-21 所示。

(6) 设置“自动套用格式”。为了使 DetailsView 控件显示美观，可以为其设置一种格式。在智能标记中选择“自动套用格式”命令，打开“自动套用格式”对话框，如图 8-22 所示，选择一种喜欢的样式。这里选择“雨天”样式。

(7) 启用编辑选项。为了使 DetailsView 控件能够编辑考生信息，在智能标记中选中“启用编辑”复选框即可，如图 8-23 所示。





图 8-19　定义参数



图 8-20　选择数据源

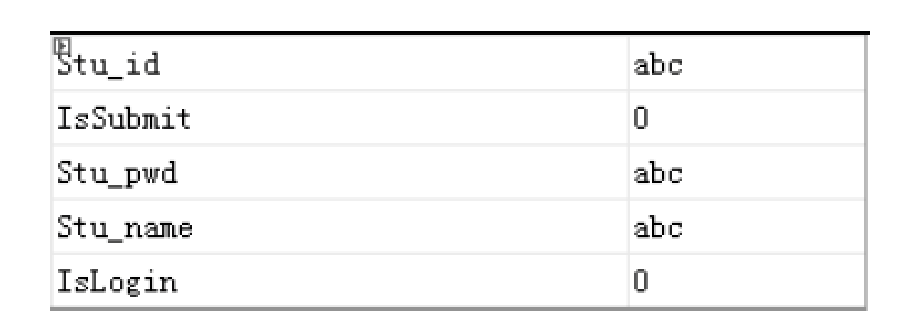


图 8-21　绑定数据源之后的 DetailsView 控件



图 8-22　设置"自动套用格式"



图 8-23　启用编辑

3．编辑字段

在 DetailsView 控件中编辑字段的方法与在 GridView 控件中编辑字段的方法类似。要编辑考生信息，必须先把绑定列转换为模板列，转换的方法为选中各个 BoundField 字段，然后选择"将此字段转换为 TemplateField"命令（如图 8-3 所示），再针对每个字段修改 HeaderText 的值，效果如图 8-24 所示。





图 8-24　DetailsView 控件的模板列

4. 班级信息的绑定

由于考生班级信息是外键字段，因此需要利用 ObjectDataSource 控件获得班级信息的数据源，并且绑定在 DetailsView 控件的“班级”字段上。具体步骤如下：

(1) 将 ObjectDataSource 控件拖入 StudentDetails.aspx 页面，并修改其 ID 值为 odsClass，然后在智能标记中选择“配置数据源”命令，打开图 8-17 所示的“配置数据源”对话框，注意业务对象选择业务逻辑层的 StuClassManager 类。

(2) 单击“下一步”按钮，进入“定义数据方法”对话框，可以设置需要的增、删、改、查方法，如图 8-18 所示，只是这里的 Select 方法要选择 GetAllStuClass()。

(3) 最后单击“完成”按钮，即可完成班级信息数据源的绑定。

(4) 编辑“班级”模板列，将在 ItemTemplate 中的 Label 控件的 DataBinding 修改为 Eval("examStuClass.Class_name")即可。

5. “编辑”功能的实现

编辑考生信息的功能主要是通过 ObjectDataSource 数据源控件中的 Update 方法实现的，但是由于“班级”信息是外键字段，因此在使用 ModifyStudentBasicInfo(String Stu_name, Int32 Class_id, String Stu_pwd, Int32 IsLogin, Int32 IsSubmit, String Stu_id)方法更新考生信息的时候，必须对它进行特殊处理。处理的方法如下：

(1) 设置“班级”字段。除了班级字段之外的其他字段的 EditTemplate 中的控件为 TextBox 控件，由于更新考生信息的时候，要随时提交班级字段信息，但是班级字段是外键，所以对班级字段进行特殊设置。设置的方法是添加 HiddenField 控件，如图 8-25 所示，并设置它的“数据绑定”值为 Eval("examStuClass.Class_id")。

(2) 添加验证。由于学号和姓名在输入的时候不能为空，因此在学号和姓名的 EditTemplate 模板中，添加非空验证控件，效果如图 8-26 所示。

(3) 在 DetailsView 控件的 ItemUpdating 事件中添加如下代码。

```
protected void dvStudentDetail_ItemUpdating(object sender, DetailsViewUpdateEventArgs e)
```



图 8-25　“班级”字段的设置

图 8-26　添加验证控件

```
{
    //将学生班级的值添加到绑定的数据源中
    this.odsStudentDetail.UpdateParameters.Add("class_id",ddlStudentClass.SelectedValue);
}
```

练　　习

1. 单项选择题

（1）GridView 控件使用________属性设置分页。

A. AllowPaging　　B. AllowSorting

C. DataSource　　D. PageIndex

（2）如果要实现某时期字段为“99-10-05”，除了要设置 DataFormatString 属性之外，还要设置________属性。

A. HtmlEncode　　B. HtmlDecode

C. UrlEncode　　D. UrEncode

（3）能够实现单向数据绑定的是________方法。

A. Eval　　B. Bind　　C. Get　　D. Set

（4）DetailsView 控件使用________属性可以设置数据的主键。

A. DefaultMode　　B. DataKey

C. DataKeyNames　　D. DataSource

2. 简答题

（1）常用的数据源控件有哪些？



（2）常用的数据绑定控件有哪些？

（3）简述 GridView 控件的字段类型。

实　　训

实训目的

（1）熟练掌握 GridView 控件的使用方法。

（2）熟练掌握 DetailsView 控件的使用方法。

实训内容

（1）新建一个网页 AdminList. aspx，利用 GridView 控件显示管理员的信息。

（2）新建一个网页 AdminDetails. aspx，利用 DetailsView 控件显示管理员的详细信息。

实训指导

（1）AdminList. aspx 页面的最终实现效果如图 8-27 所示。



图 8-27　管理员列表页面

（2）在上述页面，单击“详细”链接可以进入详细列表页面 AdminDetails. aspx，如图 8-28 所示。



图 8-28　管理员详细列表页面



任务9　主题与母版页

技能目标

会创建外观文件和为主题添加 CSS 样式的技术；能创建和绑定母版页的内容页。

知识目标

学习和掌握主题组成元素，学习和掌握母版页和内容页的基础。

任务描述

开发一些大型网站时，利用主题和母版页可以使开发人员有效地设计出外观和感觉一致的网页。而且这些 Web 网站中的页面与页面之间给人的总体外观和感觉都比较统一，非常美观。

本章任务如下：

- 为在线考试系统创建一个主题；
- 母版页的制作与使用；
- 为在线考试系统后台管理页面制作母版页。

9.1　知识准备

9.1.1　主题概述

所谓“主题”是指页面和控件外观属性设置的集合。利用主题功能，不仅可以设置页面和控件的外观，还可以在所有的 Web 应用程序、单个 Web 应用程序的所有页面或单个 Web 页面中，快速一致地应用所定义的外观。使用页面的内置主题犹如设置一个属性那么容易。主题由一个文件组构成，一般包含外观文件、级联样式表(CSS)文件、图像和其他资源，其中，外观是主题中的必要元素。

1. 外观

外观文件是主题的核心内容，它主要用于定义页面中服务器控件的外观。在主题中可以包含一个或者多个外观文件。外观文件的扩展名为.skin，其中包含对各种服务器控件(例如，Button 控件、Label 控件、TextBox 控件或 Calendar 控件)的属性设置。例如，下面是 Button 控件的外观设置代码。




```
<asp:Button runat="server" BackColor="yellow" BorderColor="blue"/>
```

说明：控件外观的代码设置与控件声明代码类似。在控件外观设置中，只能包含可作为主题的属性定义。这部分属性可能是样式属性、集合属性、模板属性或数据绑定表达式等。

一个主题可以包含一个给定控件的多个外观，每个外观都用一个唯一的名称（SkinID属性）标识。如果设置了 SkinID 属性，则称此时的外观为有名称的外观（named skin）；还有一种外观称为无名外观（即默认外观）。

例如，在相同主题中设置一个按钮的两个有名称的外观，代码如下：

```
<asp:Button  runat="server" BackColor="green"  BorderColor="yellow" SkinID="gray"/>
<asp:Button  runat="server" Height="64" Width="128" BackColor="pink"
BorderColor="red" SkinID="pink"/>
```

2. 级联样式表（CSS）

主题中可以包含一个或者多个 CSS 文件（扩展名为.css）。CSS 文件在主题功能出现之前，已经得到了广泛应用。将 CSS 文件放在主题目录中时，它将自动作为主题的一部分应用。

3. 图像和其他资源

主题中还可以包含图形和其他资源，例如，脚本文件或声音文件。通常，主题的资源文件与该主题的外观文件位于同一个文件夹中，但它们也可以存储在 Web 应用程序中的其他文件夹中。例如，若要引用主题目录的某个子文件夹中的资源文件，需要使用类似该 Image 控件外观显示的路径。代码如下：

```
<asp:Image runat="server" ImageUrl="Theme/filename.jpg"/>
```

资源文件也可以存储在主题目录以外的位置，如果将资源文件放在应用程序的 hywork 文件夹中，可以使用“～”来引用资源文件，代码如下：

```
<asp:Image runat="server" ImageUrl="~/hywork/filename.jpg"/>
```

4. 文件存储和组织方式

主题可分为全局主题和局部主题。全局主题对于 Web 服务器内的所有应用程序都可见，一般位于如下路径的子目录中：C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\ASP.NETClientFiles\Themes。局部主题也称为应用程序级主题。局部主题仅应用于单个 Web 应用程序，且必须放在应用程序根目录的 App_Themes 文件夹下。开发人员可以通过手动或者使用 Visual Studio 2005 在网站的根目录下创建该文件夹。如图 9-1 所示，在 App_Themes 文件夹中，包括两个二级文件夹“主题 1”和“主题 2”，每个主题文件夹下都可以包含外观文件、CSS 文件和图像文件等。其中外观文件是主题的核心部分，每个主题文件夹下都可以包含一个或者多个外观文件，在开发过

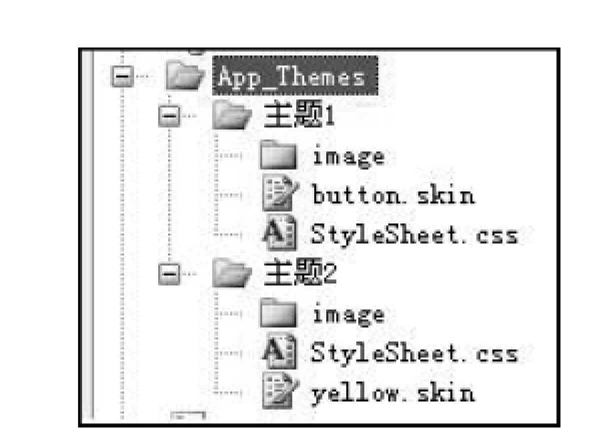


图 9-1　App_Themes 文件夹的示意图





程中可根据实际需要对外观文件进行有效管理。

下面给出 3 种常见外观文件的组织方式及其说明。

(1) 根据 SkinID 组织

在对控件进行外观设置时，将具有相同的 SkinID 的控件放在同一个外观文件中，这种方式适用于网站页面较多、设置内容复杂的情况。其文件夹结构如图 9-2 所示。

(2) 根据控件类型组织

组织外观文件时，以控件类型进行分类，这种方式适用于页面中包含控件较少的情况。其文件夹结构如图 9-3 所示。

(3) 根据文件组织

组织外观文件时，以网站中的页面进行分类，这种方式适用于网站中页面较少的情况。其文件夹结构如图 9-4 所示。



图 9-2　根据 SkinID 组织的外观文件



图 9-3　根据控件类型组织的外观文件



图 9-4　根据文件组织的外观文件

9.1.2　母版页概述

使用 ASP.NET 母版页可以为应用程序中的页创建一致的布局。单个母版页可以为应用程序中的所有页(或一组页)定义所需的外观和标准行为，然后可以创建包含要显示的内容的各个内容页。当用户请求内容页时，这些内容页与母版页合并以将母版页的布局与内容页的内容组合在一起输出。

母版页为具有扩展名 .master(如 MySite.master)的 ASP.NET 文件，它具有可以包括静态文本、HTML 元素和服务器控件的预定义布局。母版页由特殊的 @ Master 指令识别，该指令的使用使母版页有别于内容页，且每个.master 文件只能包含一条@Master 指令。

在默认的母版页中有一个容器控件，即 ContentPlaceHolder 控件，其内容占位符由一个 ID 唯一标识。例如：

```
<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
```

ContentPlaceHolder 控件本身并不包含具体内容设置，仅是一个控件声明而已。ContentPlaceHolder 控件只能在母版页中使用。如果在平常的 Web 网页发现这样一个控件，则会发生一个解析器错误。在母版页中，可以根据网页中所需的可定制区域数定义多个内容占位符，而内容页不必填充所绑定的母版页中定义的全部占位符。内容占位符可以被赋予默认内容，在内容页没有提供替代内容时，即内容页没有引用母版页中的一个给定占位



符时，则使用默认内容。但当内容页填充了占位符时，设置的默认内容将被完全忽略，且默认内容绝不会与内容页提供的标记合并。

与母版页密不可分的还有内容页，内容页是扩展名为.ascx的源文件。母版页定义了网页的架构，包含了页面的公共部分，并为可定制区域留下了占位符；内容页是只包含<asp:Content>服务标签的ASP.NET页，它的关键部分就是Content控件，可称之为内容控件，母版页中的占位符所对应的具体内容就包含在这个控件中。Content控件只能与对应的ContentPlaceHolder控件结合使用，它不是一个独立的控件。内容页与母版页的有机结合就是通过内容页中的Content控件的ContentPlaceHolderID属性绑定到母版页中的ContentPlaceHolder控件的ID属性实现的，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

内容页只能包含Content控件，并且在<asp:Content>外是不允许有服务器控件或文本文字的。同时，在一个内容页中可以有一个或多个Content控件，但它们必须指向母版中的不同占位符。

9.1.3　母版页的制作

在ASP.NET 2.0中，除了@Master指令和一个或多个ContentPlaceHolder服务器控件外，母版页基本上类似于标准的ASP.NET页。例如，二者在代码结构方面都需要声明<html>、<body>、<form>以及其他Web元素等。唯一重大区别就是在母版页中使用的是ContentPlaceHolder容器控件。但在母版页中包含的是页面的公共部分，因此在创建母版页之前，必须判断哪些内容是页面的公共部分。使用Visual Studio 2005创建母版页的具体操作步骤如下：

(1) 打开Visual Studio 2005，新建一个ASP.NET网站，编程语言采用C#。

(2) 在解决方案资源管理器中，右击网站名称，在弹出的快捷菜单中选择“添加新项”命令，如图9-5所示。



图9-5　添加新项

(3) 打开“添加新项”对话框，如图9-6所示。在“模板”选项区域中选择“母版页”选项，在“名称”文本框中输入“MasterPage.master”。

(4) 单击“添加”按钮，新建的母版页就会被添加到解决方案资源管理器中，如图9-7所示。

(5) 在Visual Studio 2005视图模式下，可看到创建的默认的母版页。还可以添加多个ContentPlaceHolder控件。







图 9-6　“添加新项”对话框



图 9-7　添加 MasterPage.master

9.2　任务实施

9.2.1　添加在线考试系统主题

在线考试系统主题的添加步骤如下：

(1) 创建主题文件夹。在解决方案资源管理器中站点名称上右击，从弹出的快捷菜单中选择“新建文件夹”命令，创建一个名为 App_Themes 的文件夹。利用同样的步骤在该文件夹下再创建一个名为 mytheme 的子文件夹。

(2) 添加外观文件。右击子文件夹 mytheme，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，如图 9-8 所示。在“模板”选项区域中选中“外观文件”选项，并在“名称”文本框中输入“OnLineSkin.skin”，然后单击“添加”按钮，弹出确认窗口如图 9-9 所示。单击“是”按钮，此时就在解决方案资源管理器中添加了 App_Themes 文件夹和 OnLineSkin.skin 文件，如图 9-10 所示。

(3) 在 OnLineSkin.skin 外观文件中添加相关代码，用来设置页面中各种控件的外观。在下面代码中分别创建了 TextBox 控件、Button 控件、GridView 控件和 DropDownList 控





图 9-8 “添加新项”对话框

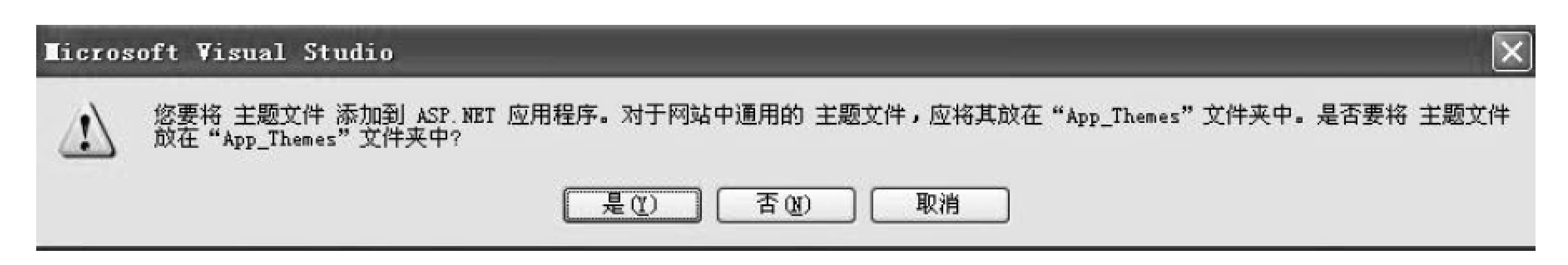


图 9-9 确认窗口



图 9-10 添加完成后文件样式

件的外观。OnLineSkin.skin 外观文件的源代码如下：

```
<asp:DropDownList  runat="server" BackColor="#C0FFFF" ForeColor="#C00000"
    Height="62px" Width="126px">
</asp:DropDownList>
<asp:TextBox  runat="server" BackColor="#C0FFFF" BorderColor="Navy"
    BorderStyle="Solid" Height="20px" Width="120px"></asp:TextBox>
<asp:Button  runat="server" BackColor="#C0FFFF" BorderColor="#C00000"
    BorderStyle="Outset" Height="28px" Width="120px"/>
<asp:GridView  runat="server" BackColor="#C0FFFF" BorderColor="Navy" BorderStyle="Solid"
BorderWidth="1px">
```



```
</asp:GridView>
<asp:DetailsView  runat="server" BackColor="#C0FFFF" BorderColor="Navy"
    BorderStyle="Solid" BorderWidth="1px" Height="50px" Width="125px">
</asp:DetailsView>
```

代码添加有一个小的技巧：添加一个页面，从工具箱中将这些控件拖入设计页面中，然后在属性窗口中分别设置这些控件的外观属性，再在源视图中把代码复制出来修改一下就行了，即把控件的ID属性设置去掉。

(4) 在StudentLogin.aspx页面中应用该主题，该页面的源代码添加如图9-11所示。在页面中添加Theme属性并设置为OnLineSkin。

```
<%@ Page Language="C#" AutoEventWireup="true" Theme ="OnLineSkin" CodeFile="StudentLogin.aspx.cs" Inherits=
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tran
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>学生登录考试</title>
```

图9-11　在页面中应用主题

(5) 应用主题后StudentLogin.aspx页面的效果如图9-12所示。



图9-12　主题应用后的效果

(6) 为主题添加CSS样式。在OnLineSkin文件夹上右击，从弹出的快捷菜单中选择“添加新项”命令，在弹出的“添加新项”对话框中选择“样式表”选项，将名称设置为OnLineStyle.css。源代码如下：

```
body
{
    text-align: center;
    color :Red ;
    background-color: White;
}
A:link
{
    color: Blue;
    text-decoration: underline;
}
A:visited
```



```
{
    color: Blue;
    text- decoration: underline;
}
A:hover
{
    color: Navy;
     text- decoration: none;
}
INPUT
{
    background- color: Aqua;
}
```

(7) 应用 OnLineStyle.css 样式后的 StudentLogin.aspx 页面效果如图 9-13 所示。

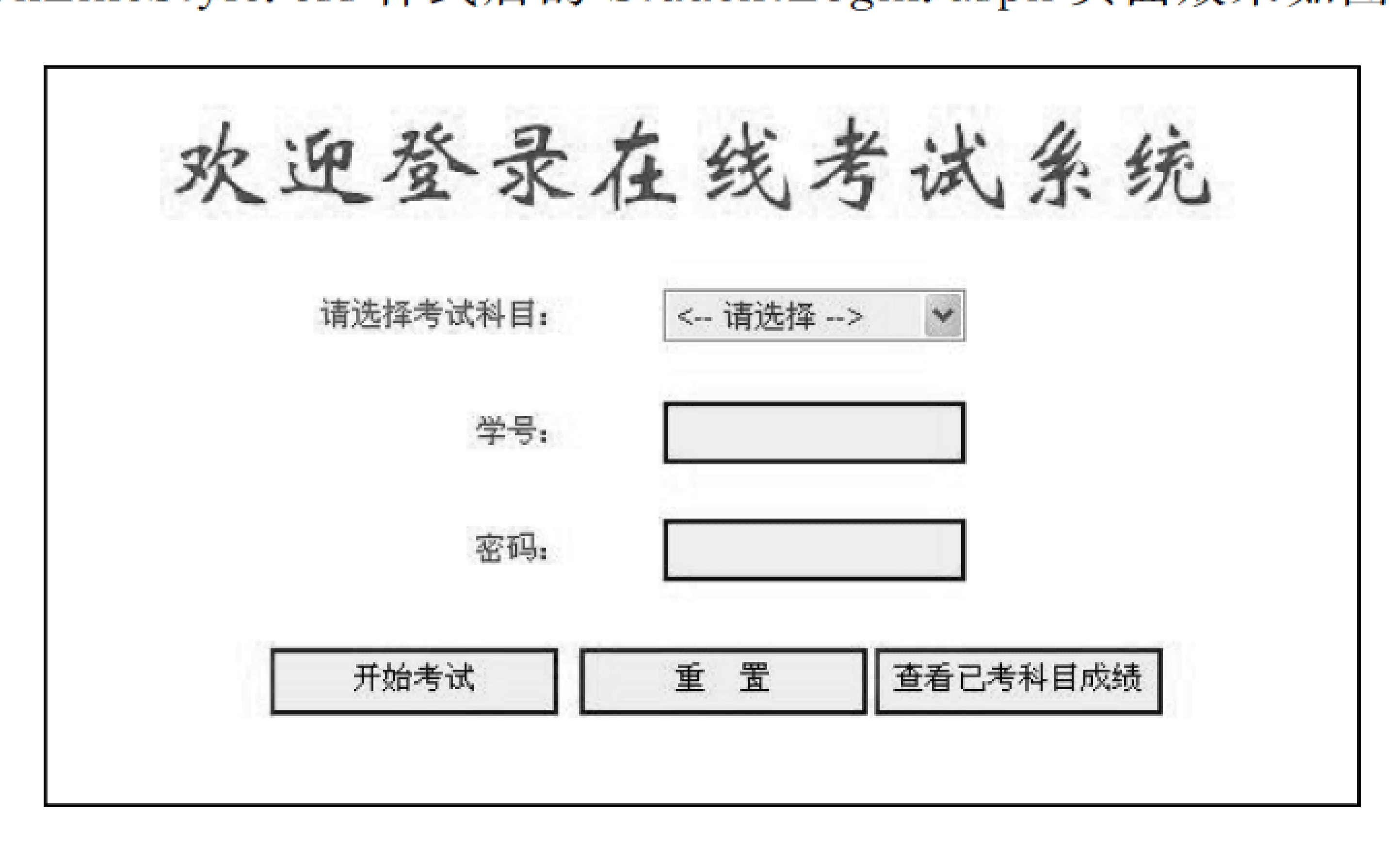


图 9-13　为主题添加 CSS 样式后

9.2.2　母版页的套用

在创建完一个完整的母版页之后,接下来就必然是创建内容页,用来套用母版页。内容页的创建与母版页的创建相似,其创建过程比较简单。创建内容页的步骤如下:

(1) 在解决方案资源管理器中,右击网站名称,在弹出的快捷菜单中选择"添加新项"命令。

(2) 打开"添加新项"对话框,如图 9-14 所示,要求选择新建文件类型。由于内容页与普通.aspx 页面的扩展名相同,因此,在"模板"选项区域中选择"Web 窗体"选项,在"名称"文本框中将其命名为"Default.aspx",选中"将代码放在单独的文件中"和"选择母版页"(要创建内容页必须选中此项,内容页必须绑定到母版页才有实际意义)两个复选框。

(3) 单击"添加"按钮,将打开"选择母版页"对话框,如图 9-15 所示,在"选择母版页"对话框中,窗口左侧是项目文件夹,右侧是文件夹中的母版页列表。此时母版页列表中只有一个母版页 MasterPage.master 文件。选中该文件,单击"确定"按钮,即可完成一个绑定到母版页的内容页 Default.aspx 了。





图 9-14　添加窗体应用母版页



图 9-15　选择要应用的母版页

9.2.3　为在线考试系统后台管理页面制作母版

任何一个网站中的页面都有其相同之处(比如 banner 信息、导航信息和版权信息),为了便于制作和管理,常常把相同的内容做到母版里面。下面为在线考试系统后台管理页面制作母版页 AdminMaster.master,该页面设计的效果如图 9-16 所示。

具体设计步骤如下:

(1) 在解决方案资源管理器中,右击"D:/OnLine/OnLineWeb"项目下的 admin 文件夹(因为此模板只用于后台管理,所以放在 admin 文件夹下),在弹出的快捷菜单中选择"添加新项"命令,如图 9-17 所示。

(2) 在出现的"添加新项"对话框中,在"模板"选项区域中选择"母版页"选项,在"名称"





图 9-16　在线考试系统后台管理母版页



图 9-17　“添加新项”命令

文本框中输入“MasterPage.master”，如图 9-18 所示。单击“添加”按钮，新建的母版页就会被添加到解决方案资源管理器中，如图 9-19 所示。



图 9-18　“添加新项”对话框

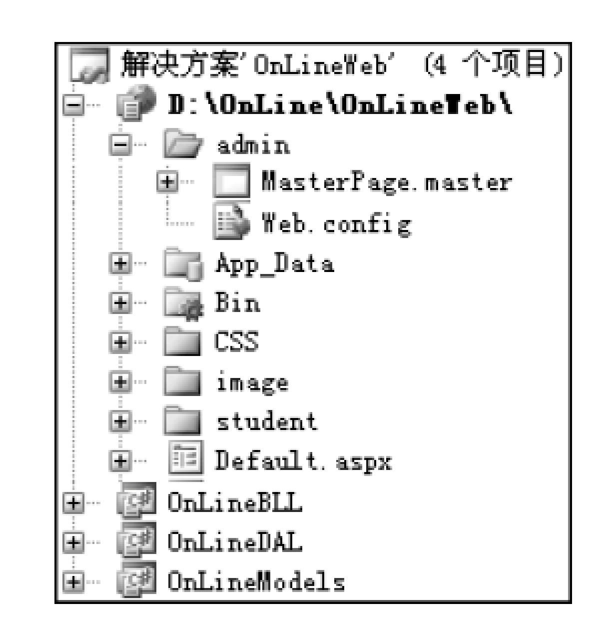


图 9-19　添加 MasterPage.master

(3) 在 Visual Studio 2005 源模式下，可看到创建的自动生成的代码如下：

```
<%@ Master Language="VB" CodeFile="MasterPage2.master.vb" Inherits="OnLineWeb_
MasterPage2" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>无标题页</title>
</head>
<body>
```



```
        <form id="form1" runat="server">
        <div>
            <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
            </asp:contentplaceholder>
        </div>
        </form>
    </body>
    </html>
```

（4）在 Visual Studio 2005 设计模式下，添加 Table 控件，来设置母版页的 top、content 和 foot 部分。具体设置属性如下：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="AdminMaster.master.cs"
Inherits="admin_AdminMaster" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>管理员</title>
    <link href="../Css/main.css" rel="stylesheet" type="text/css">
</head>
<body>
    <form id="form1" runat="server">
<div id="top">
<table  width="950" border="0" cellpadding="0" cellspacing="0">
    <tr>
        <td align="center"colspan="2" bgcolor="#73AFF2"><img src="../image/top.gif"/></
        td>
    </tr>
</table>
</div>
  <div id="content">
  <table border="0" width="950">
       <tr>
        <td width="160" valign="top" bgcolor="#559DFE"><table width="160">
           <tr>
             <td align="center" style="width: 188px; font-weight: bold; font-size: 18px; color:
             teal;">
                 后台管理</td>
          </tr>
          <tr>
             <td align="center"   style="width: 188px">
             <table><tr>
             <td>  </td><td width="100px">
              <div align="left">
             <asp:TreeView ID="TreeView1" runat="server" DataSourceID="xdsLZZX"   ImageSet
               ="Arrows" Width="100px">
                        <DataBindings>
                          <asp:TreeNodeBinding DataMember="siteRoot" TextField="title"/>
                               <asp:TreeNodeBinding DataMember="siteMapNode" NavigateUrlField=
                                "url" TextField="title"/>
```



```
                    </DataBindings>
                <ParentNodeStyle Font-Bold="False"/>
                <HoverNodeStyle Font-Underline="True" ForeColor="#5555DD"/>
                <SelectedNodeStyle Font-Underline="True" ForeColor="#5555DD"
                 HorizontalPadding="0px"
                    VerticalPadding="0px"/>
                    < NodeStyle  Font - Names = " Verdana "  Font - Size = " 10pt "  ForeColor =
                     "Black" HorizontalPadding="5px"
                    NodeSpacing="0px" VerticalPadding="3px"/>
                </asp:TreeView></div>
                <asp:XmlDataSource ID="xdsLZZX" runat="server" DataFile=
                "~/admin/admin_menu.xml" XPath="/*"></asp:XmlDataSource>
                </td></tr></table>
            </td>
          </tr>
        <tr>
            <td style="height: 5px; width: 160px;" colspan="2"></td>
        </tr>
        </table>
        </td>
        <td width="100%" rowspan="2" valign="top">
          <table width="100%" >
            <tr>
              <td  align="left"><strong>您的位置:<asp:SiteMapPath ID="SiteMapPath1"
              runat="server">
                  </asp:SiteMapPath>
              </strong></td>
             </tr>
          </table>
        <asp:contentplaceholder id="cphAdmin" runat="server">
        </asp:contentplaceholder>
        </td>
      </tr>
</table>
</div>
<div id="foot">
    <table border="0" width="950" bgcolor="#559DFE">
        <tr>
      <td height="24" ><div align="center">copyright&copy;2009 清华大学出版社
      </div><hr></td>
    </tr>
    <tr style="height:5px">
      <td  ></td>
    </tr>
    </table>
    </div>
    </form>
</body>
</html>
```



练　　习

简答题

使用主题和使用 CSS 文件有什么不同？

实　　训

实训目的

（1）熟练掌握模板页的制作与套用。
（2）为在线考试前台页面制作母版页。

实训内容

为在线考试系统的前台制作一个母版页 MasterPage.master。

实训指导

（1）添加页面 MasterPage.master，步骤与前面 AdminMaster.master 页面的添加相同。

（2）添加 table 控件来布局，界面设计效果如图 9-20 所示。



图 9-20　在线考试系统前台母版页

任务10　DataList控件与试题信息管理



技能目标

能使用 DataList 控件分页显示试题信息,删除试题信息;能编写代码实现对试题信息排序。

知识目标

掌握 DataList 数据展示控件的特点,掌握 DataList 控件的使用;理解并掌握 PagedDataSource 类的使用。

任务描述

本任务主要介绍了 DataList 控件的基本用法,同时介绍了如何使用该控件分页显示试题信息并进行排序。

本章任务如下:

- 试题列表页面的展示;
- 试题列表页面的分页;
- 试题列表页面的排序;
- 试题的删除。

10.1　知识准备

10.1.1　DataList 控件

1. DataList 控件介绍

一般而言,对于多行多列数据(或称为表格类数据)会选择前面讲过的 GridView 控件来展示。对于单行多列或多行单列的数据,则建议使用 DataList 和 Repeater 控件。DataList 控件提供了一些简单的模板。如果需要精确控制布局,可以考虑使用 Repeater 控件,该控件将在后面章节中讲解。

2. DataList 控件的使用

DataList 控件可以使用模板和样式对数据的显示样式进行定义,可用于创建模板化的列表数据,可以在一个模板化列表中,通过设置模板来控制呈现该列表的 HTML。模板描述了如何显示列表中的某一项的 HTML,比如可以显示诸如一行中有多列的内容,此时可用于任

何重复结构中的数据。图 10-1 给出了试题列表页面，后文将详细描述该页面的制作过程。

科目	试题类型	试题答案	试题描述	详细	删除
计算机基础	单选	B	世界上公认的第一台计算机是_____。	详细	删除
计算机基础	单选	A	计算机的发展从结构和功能等方面看，大致有微型化、网络化、多媒体化和...	详细	删除
计算机基础	单选	D	第4代计算机的主要元器件采用的是_____。	详细	删除
计算机基础	单选	B	计算机内存的每个基本单元都被赋予一个唯一的序号，称为_____。	详细	删除
计算机基础	单选	C	在个人计算机系统中，打印机一般是通过_____接口连接到系统总线上...	详细	删除
计算机基础	单选	C	微型计算机（PC）的文件系统采用的是_____结构。	详细	删除
计算机基础	单选	C	计算机的存储单元中存储的内容_____。	详细	删除
计算机基础	单选	D	把内存中的数据传送到计算机的硬盘，称为_____。	详细	删除

图 10-1　试题列表

通过分析该页面，可以将这个页面看成是多行单列的表格，每行表示一道试题，都包含该试题的所属科目、试题类型、试题答案、试题描述、详细信息等内容。所以，使用 DataList 时必须定义一个 ItemTemplate（项模板）；另外，还有一些可选模板可用于定制列表的外观。

显然它不像 GridView 控件那样直接显示一个表格，而是需要编辑模板列。和 GridView 一样，可以通过右键菜单打开模板编辑界面，如图 10-2 所示。其中可使用的模板比 GridView 要少，不过我们仅用来展示页面，这些模板已经足够了，而且事实上，在这里仅使用 ItemTemplate（项模板）和 SeparatorTemplate（分隔符模板）就可以了。

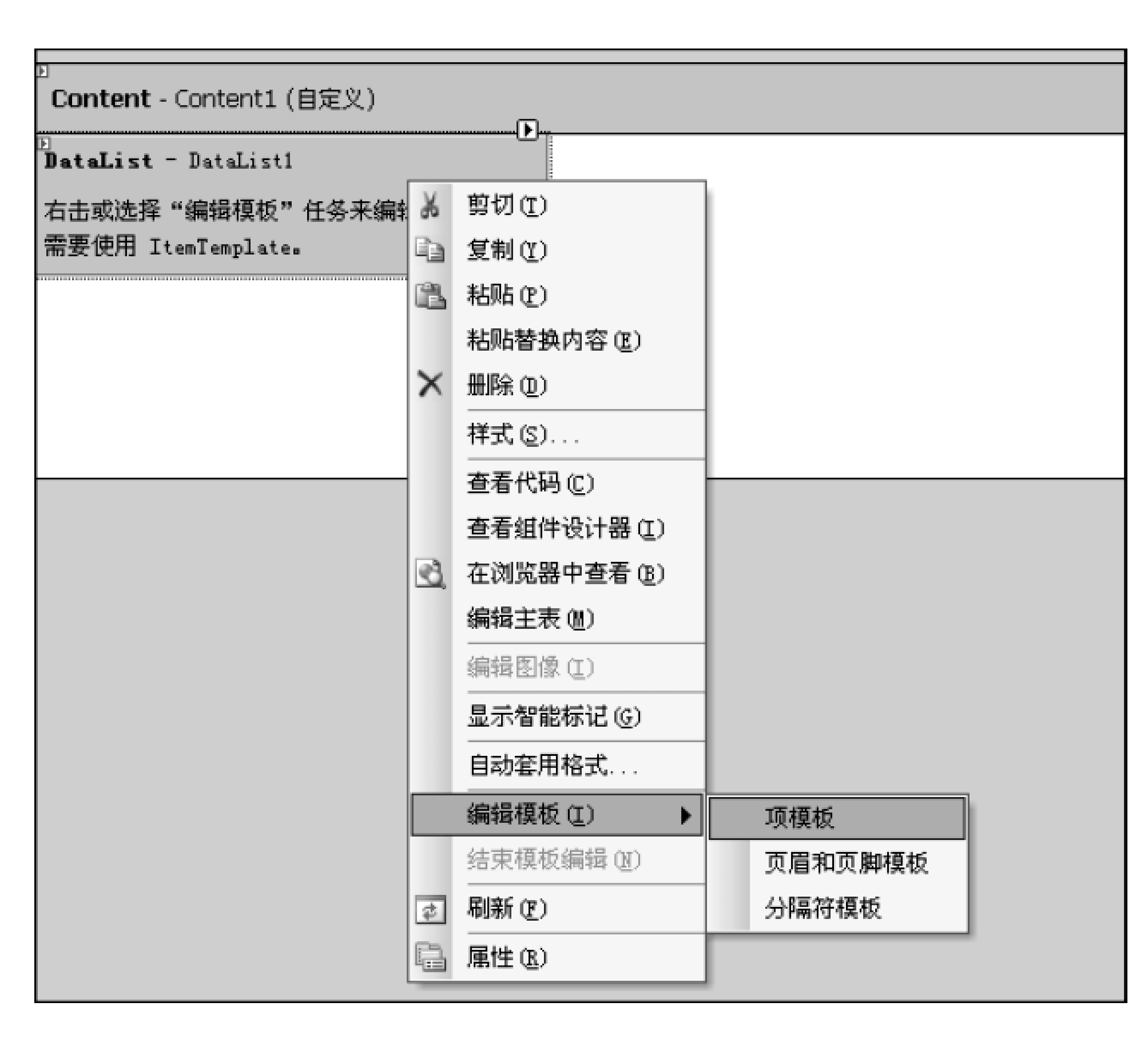


图 10-2　DataList 控件的模板

表 10-1 所示出了 DataList 控件的模板列表。



表 10-1　DataList 控件的模板列表

模 板 名 称	说　明
ItemTemplate	项模板,标记了每行显示的内容
AlternatingItemTemplate	交替项模板,可设置交替行显示不同的风格,不设置时,与项模板相同
SelectedItemTemplate	选中项模板,设置选中后的特殊样式
EditItemTemplate	编辑项模板
HeaderTemplate 和 FooterTemplate	页眉和页脚的模板
SeparatorTemplate	分隔符模板,一般用＜hr＞(一条直线)

10.1.2　DataList 控件的分页和排序

试题的数量总是很多，因此需要增加分页功能;另外,对这些试题进行排序也是很必要的。实现排序和分页后的页面如图 10-3 所示。但是 DataList 控件没有内置分页排序的功能,所以要在后台编码实现分页和排序的效果。

排序方式： 试题类型　科目

科目	试题类型	试题答案	试题描述	详细	删除
计算机基础	单选	B	世界上公认的第一台计算机是_____。	详细	删除
计算机基础	单选	A	计算机的发展从结构和功能等方面看，大致有微型化、网络化、多媒体化和...	详细	删除
计算机基础	单选	D	第4代计算机的主要元器件采用的是_____。	详细	删除
计算机基础	单选	B	计算机内存的每个基本单元都被赋予一个唯一的序号，称为_____。	详细	删除
计算机基础	单选	C	在个人计算机系统中，打印机一般是通过_____接口连接到系统总线上...	详细	删除
计算机基础	单选	C	微型计算机（PC）的文件系统采用的是_____结构。	详细	删除
计算机基础	单选	C	计算机的存储单元中存储的内容_____。	详细	删除
计算机基础	单选	D	把内存中的数据传送到计算机的硬盘，称为_____。	详细	删除

第 1 页 共 42 页　上一页　下一页

图 10-3　实现排序和分页后的页面

1. DataList 控件的分页

DataList 控件的分页可以使用分页类 PagedDataSource 来实现,该类封装了数据绑定控件与分页相关的属性。PagedDataSource 类的常用属性如表 10-2 所示。

只要将数据源和当前页数赋值给 PagedDataSource 类的实例对象,其他属性(如总记录数和总页数)可以自动计算得出。



表 10-2　PagedDataSource 类的常用属性

属　　性	说　　明	属　　性	说　　明
CurrentPageIndex	当前页	PageSize	每页记录数
PageCount	总页数	AllowPaging	控件是否实现自动分页功能
Count	总记录数	DataSource	数据源

我们编写的绑定方法如下：

```
private void Databind()
{
    PagedDataSource pdsQuestions=new PagedDataSource();
    //对 PagedDataSource 对象的相关属性赋值
    pdsQuestions.DataSource=AllQuestionsManager.GetAllQuestionBySql((string)
    ViewState["OrderType"]);
    pdsQuestions.AllowPaging=true;                          //允许分页
    pdsQuestions.PageSize=8;                                //页的大小
    pdsQuestions.CurrentPageIndex=Pager;
    lblCurrentPage.Text ="第"+ (pdsQuestions.CurrentPageIndex+1).ToString()+"页 共"+
    pdsQuestions.PageCount.ToString()+"页";
    SetEnable(pdsQuestions);
    //把 PagedDataSource 对象赋给 DataList 控件
    dlQuestions.DataSource=pdsQuestions;
    dlQuestions.DataBind();
}
```

可以看到，PagedDataSource 用起来还是比较方便的。其使用过程如下：

（1）指定 PagedDataSource 实例对象的数据源为 GetAllQuestionBySql 返回的数据的集合。

（2）分别设置允许分页（AllowPaging）、页大小（PageSize）、当前页（CurrentPageIndex）的属性。

（3）指定数据展示控件的数据源为该 PagedDataSource 实例对象，并绑定。

2．DataList 控件的排序

要实现排序，最直接的办法就是在数据库端就排好顺序，所以我们可以在数据访问层访问数据时，直接添加排序条件。当然，页面数据在绑定时，需要通过业务逻辑层传递相关的参数。

首先编写数据访问层的获得已排序结果集的方法，代码如下：

```
public static IList<examAllQuestion> GetAllQuestionBySql(string orderType)
{
    string sql="select * from exam_allQuestions";
    if(orderType.Trim().Length>0)
    {
      sql+="order by"+orderType;
    }
      return GetQuestionBySql(sql);
}
```

该方法接收业务逻辑层传递过来的排序条件 orderType，根据该排序条件对相应的



SQL 语句做出修改。

下面的方法根据 SQL 语句将排序后数据库中的试题信息存放在集合 questionList 中。

```
private static IList<examAllQuestion> GetQuestionBySql(string questionSql)
{
    List<examAllQuestion> questionList=new List<examAllQuestion>();
    //使用 using 指令及时释放资源
    using (DataTable table=ConnDBHelper.GetDataSet(questionSql))
    {
    foreach (DataRow row in table.Rows)
    {
        examAllQuestion examQuestion=new examAllQuestion();
        examQuestion.Question_id=(int)row["question_id"];
        examQuestion.Question_subject=(string)row["question_subject"];
        examQuestion.Question_keys=(string)row["question_keys"];
        examQuestion.A=Convert.ToString(row["a"]);
        examQuestion.B=Convert.ToString(row["b"]);
        examQuestion.C=Convert.ToString(row["c"]);
        examQuestion.D=Convert.ToString(row["d"]);
        examQuestion.examAllSubject=AllSubjectService.GetSubjectById((int)row
        ["subject_id"]);                                                    //外键
        examQuestion.examQuestionType=QuestionTypeService.GetQuestionTypeById
        ((int)row["question_type"]);                                        //外键
        questionList.Add(examQuestion);
    }
    return questionList;
    }
}
```

上述这两种方法都编写在 AllQuestionsService 类中，这样，只要在业务逻辑层编写相关的方法传递参数就可以了。业务逻辑层的 AllQuestionsManager 类中，添加方法如下：

```
public static IList<examAllQuestion> GetAllQuestionBySql(string orderType)
{
    return AllQuestionsService.GetAllQuestionBySql(orderType);
}
```

这样，在表示层做数据绑定的时候，可以传递参数获得排序的结果了。

3. 保持页面和条件信息

现在就可以编写方法实现分页和排序了，不过还有一个问题就是状态保持，当前页数和排序条件需要进行状态保持。

因为该分页和排序信息仅需要在该页面有效，所以前面学过的 Session、Cookie、Application 的状态保持方式并不适合。

在 ASP. NET 中，有一个页面级的状态保持——ViewState。使用页面级状态保持的好处就是不影响其他页面的分页。ViewState 的用法和 Session 的用法一样。

```
ViewState["名称"]=值;
```



比如，要把页数“0”的信息存入 ViewState 就可以这样写：

```
ViewState["page"]=0;
```

10.2　任务实施

10.2.1　使用 DataList 控件分页显示试题信息

1. 设计 QuestionLists.aspx 页面布局

首先，设计 QuestionLists.aspx 页面的布局，设计步骤如下：

（1）在项目的 admin 文件夹上右击，从弹出的快捷菜单中选择“添加新项”命令，在出现的“添加新项”对话框中，把页面名称命名为 QuestionLists.aspx，在“语言”下拉列表框中选择 Visual C# 选项，选中“选择母版页”和“将代码放在单独的文件中”复选框，如图 10-4 所示。单击“添加”按钮，打开“选择母版页”对话框，选择 admin 文件夹中的 AdminMaster.master 母版页，如图 10-5 所示。

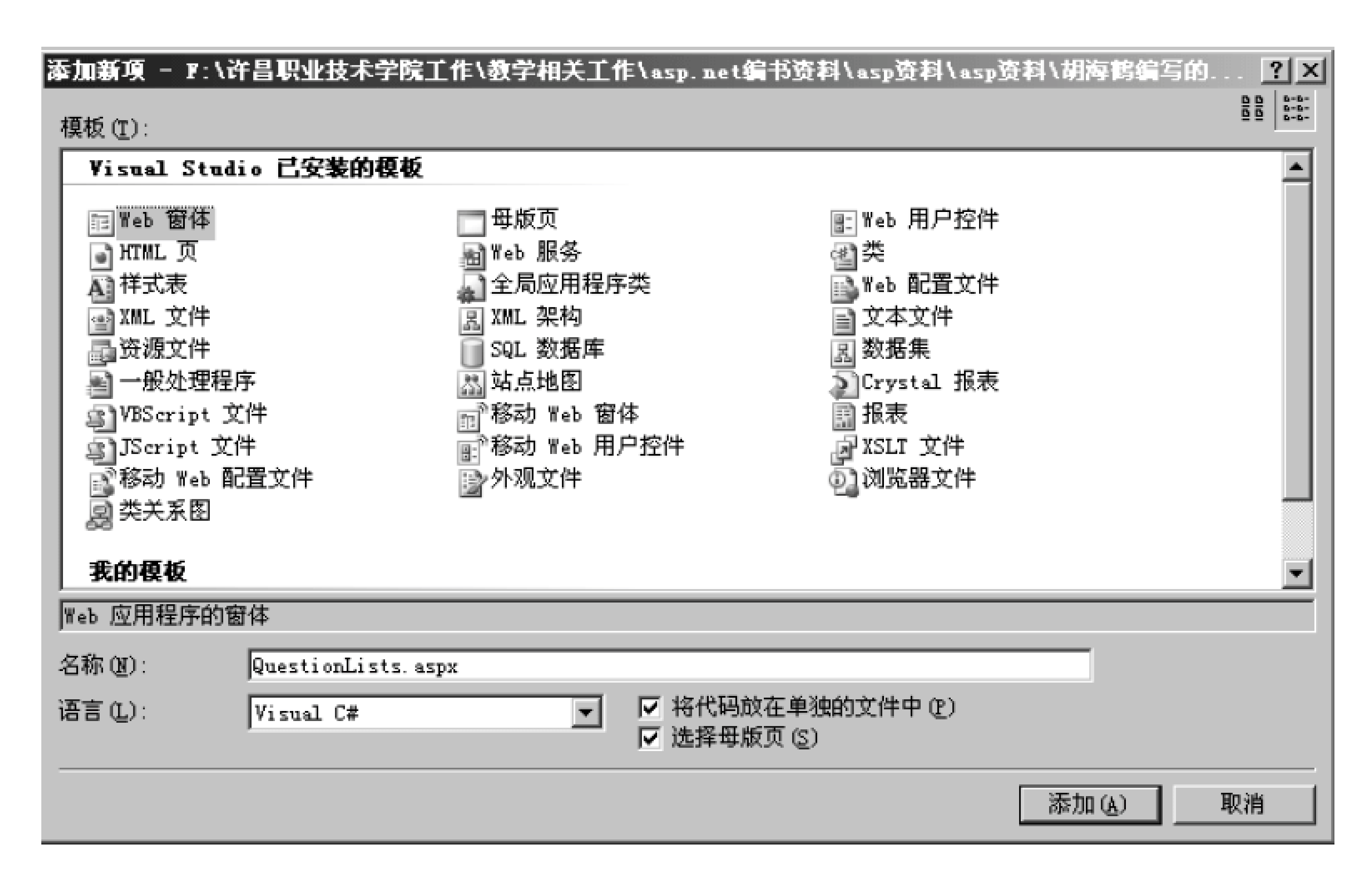


图 10-4　添加 QuestionLists.aspx 页面

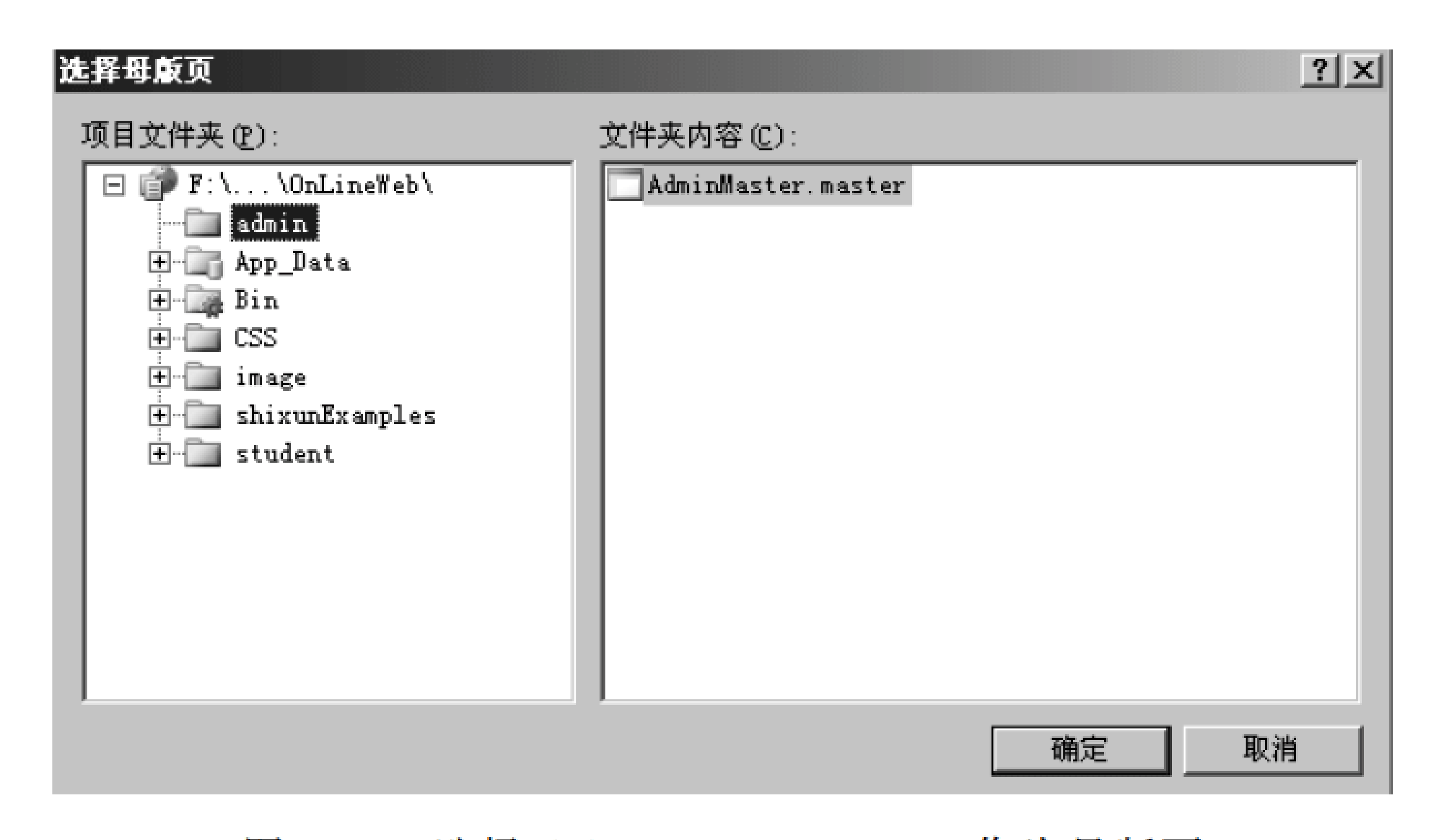


图 10-5　选择 AdminMaster.master 作为母版页



(2) 在 QuestionLists. aspx 页面的内容页中添加 DataList 控件,并设置其 ID 为 dlQuestions,如图 10-6 所示。



图 10-6　添加 DataList 控件

(3) 在 DataList 控件的项模板中添加一个表格,该表格的每一列用于显示试题的相关信息,如图 10-7 所示。该表格的设置见如下代码中斜体部分。由此可知,将每一列中需要替换的部分,直接替换为绑定语句即可。

数据绑定	数据绑定	数据绑定	数据绑定	详细	删除
数据绑定	数据绑定	数据绑定	数据绑定	详细	删除
数据绑定	数据绑定	数据绑定	数据绑定	详细	删除
数据绑定	数据绑定	数据绑定	数据绑定	详细	删除
数据绑定	数据绑定	数据绑定	数据绑定	详细	删除

图 10-7　DataList 控件项模板中显示的表格

```
<asp:DataList ID="dlQuestions"runat="server">
      <ItemTemplate>
        <table style="width:776px">
            <tr align="center"height="25">
              <td style="width:88px"><%#Eval("examAllSubject.Subject_name")%></td>
              <td style="width:67px"><%#Eval("examQuestionType.Type_name")%></td>
              <td style="width:67px"><%#Eval("Question_keys")%></td>
              <td width=""><a href="QuestionDetails.aspx?qId=<%#Eval("question_id")%>">
             <%#GetCut(Eval("Question_subject").ToString(),35)%></a></td>
                  <td width="40px"><a href="QuestionDetails.aspx?qId=<%#Eval
                  ("question_id")%>">详细</a></td>
                  <td width="40px"><a href="DeleteQuestion.aspx?qId=<%#Eval
                  ("question_id")%>">删除</a></td>
          </tr>
        </table>
      </ItemTemplate>
      <SeparatorTemplate>
```



```
        <hr/>
        </SeparatorTemplate>
    </asp:DataList>  
```

(4) 在 DataList 控件上方放置一表格，用以显示 DataList 控件中表的表头信息，如图 10-8 所示。

科目	试题类型	试题答案	试题描述	详细	删除

图 10-8　表头信息

经过上述 4 个步骤，就把显示试题列表的页面 QuestionLists. aspx 设置好了，如图 10-9 所示。

图 10-9　QuestionLists. aspx 页面

由于需要实现分页和排序功能，所以在上面的页面中需要增加按"试题类型"和"科目"排序的按钮，并且增加页面切换的"上一页"和"下一页"按钮。页面效果如图 10-10 所示。

图 10-10　在 QuestionLists. aspx 页面增加"上一页"和"下一页"按钮

分页和排序的页面代码如下：

```
<div class="contentstyle">
```



```
    <div id="divOrder">
      <div style="text-align:left;margin:20px 0 20px 0;">排序方式:
      <asp:Button ID="btnType" runat="server" Text="试题类型"
        OnClick="btnType_Click" BackColor="#99ccff" BorderColor="DarkGray" BorderStyle=
        "Solid" BorderWidth="1px" CssClass="anniu" FontSize="12px"
        ForeColor="Black" FontBold="False" Height="16px" Width="57px"/>
      <asp:Button ID="btnSubject" runat="server" Text="科目" BackColor="#99ccff"
        BorderColor="DarkGray" BorderStyle="Solid"
        BorderWidth="1px" CssClass="anniu" FontSize="12px" ForeColor="Black"
        FontBold="False" Height="16px" Width="57px" OnClick="btnSubject_Click"/>
      </div>
    </div>
</div>
```

2. 使用 DataList 控件分页显示试题信息

实现分页时，在 QuestionLists.aspx.cs 中编写如下代码。

（1）页面加载时的方法

```
protected void Page_Load(object sender,EventArgs e)
{
    if (!IsPostBack)
    {
        //首次加载,赋初值
        ViewState["Page"]=0;
        ViewState["OrderType"]="";
        DataBind();
    }
}
```

此方法中，Page 页用于保持当前页数，OrderType 用于保持排序条件。在页面加载时，对 ViewState 所保持的状态赋初值，则在使用时不需要再作特殊的判断（比如非空验证）。

（2）数据绑定方法

```
private void DataBind()
{
    PagedDataSource pdsQuestions=new PagedDataSource();
    //对 PagedDataSource 对象的相关属性赋值
    pdsQuestions.DataSource=AllQuestionsManager.GetAllQuestionBySql((string)ViewState["
    OrderType"]);
    pdsQuestions.AllowPaging=true;
    pdsQuestions.PageSize=8;
    pdsQuestions.CurrentPageIndex=Pager;
      lblCurrentPage.Text="第"+(pdsQuestions.CurrentPageIndex+1).ToString()+
      "页 共"+pdsQuestions.PageCount.ToString()+"页";
    SetEnable(pdsQuestions);
    //把 PagedDataSource 对象赋给 DataList 控件
    dlQuestions.DataSource=pdsQuestions;
    dlQuestions.DataBind();
}
```





此方法中的 SetEnable(pdsQuestions)方法用于设置上下页按钮的有效状态,即:浏览第一页时,设置"上一页"按钮无效;当浏览最后一页时,设置"下一页"按钮无效。

(3) SetEnable(pdsQuestions)方法

```
private void SetEnable(PagedDataSource pds)
{
    btnPrev.Enabled=true;
    btnNext.Enabled=true;
    if (pds.IsFirstPage)
          btnPrev.Enabled=false;
    if (pds.IsLastPage)
          btnNext.Enabled=false;
}
```

(4) 关于"下一页"、"上一页"按钮的方法

```
protected void btnNext_Click(object sender, EventArgs e)
{
    Pager++;
    DataBind();
}
protected void btnPrev_Click(object sender, EventArgs e)
{
    Pager--;
    DataBind();
}
```

在这两个方法中,要注意调用方法绑定数据。

(5) 定义一个属性

```
private int Pager
{
    get
    {
        return (int)ViewState["Page"];
    }
    set
    {
        ViewState["Page"]=value;
    }
}
```

此处定义 Pager 属性的作用是:将当前页数作为本页的属性,使用起来比较方便。

通过上述方法,便实现了试题的分页显示。

10.2.2 删除试题信息

1. 思路

试题删除时,可以在 admin 文件夹中新建一个页面 DeleteQuestion.aspx,把要删除试



题的 ID 号作为参数传递给 DeleteQuestion.aspx 页面，就可以实现试题删除。

2. 数据访问层添加的方法

在数据访问层的 AllQuestionsService 类中的代码如下：

```
public static void DeleteQuestionById(int questionId)
{
    string sql="delete exam_allQuestions where question_id=@questionId";
    try
    {
        SqlParameter[] para=new SqlParameter[]
        {
            new SqlParameter("@ questionId", questionId)
        }
        ConnDBHelper.ExecuteCommand(sql, para);
    }
  catch (Exception e)
  {
        Console.WriteLine(e.Message);
        throw e;
  }
}
```

3. 业务逻辑层添加的方法

在业务逻辑层的 AllQuestionsManager 类中的代码如下：

```
public static void DeleteQuestionById(int Question_Id)
{
    AllQuestionsService.DeleteQuestionById(Question_Id);
}
```

4. 用户界面表示层添加的方法

DeleteQuestion.aspx.cs 中的代码如下：

```
protected void Page_Load(object sender,EventArgs e)
{
    string queId=Request.QueryString["qId"];
    if (queId !=null)
    {
        Response.Write("<script>alert('确定删除此项吗!');
        window.location='QuestionLists.aspx'</script>");
        AllQuestionsManager.DeleteQuestionById(Convert.ToInt32(queId));
        //Response.Redirect("QuestionLists.aspx");
    }
}
```

通过上述在数据访问层、业务逻辑层、用户界面表示层的三种方法，实现试题的删除。



10.2.3 对试题信息排序

1. 思路

在数据库端就排好顺序，所以在数据访问层访问数据时，直接添加排序条件，页面数据在绑定时，需要通过业务逻辑层传递相关的参数。

2. 数据访问层添加的方法

实现排序时，首先编写数据访问层的 AllQuestionsService 类的获得已排序结果集的方法，代码如下：

```
public static IList<examAllQuestion>GetAllQuestionBySql(string orderType)
{
    string sql="select* from exam_allQuestions";
    if(orderType.Trim().Length>0)
    {
        sql+="order by"+orderType;
    }
    return GetQuestionBySql(sql);
}
private static IList<examAllQuestion>GetQuestionBySql(string questionSql)
{
    List<examAllQuestion>questionList=new List<examAllQuestion>();
    //使用 using 指令及时释放资源
    using (DataTable table=ConnDBHelper.GetDataSet(questionSql))
    {
        foreach(DataRow row in table.Rows)
    {
        examAllQuestion examQuestion =new examAllQuestion();
        examQuestion.Question_id = (int)row["question_id"];
        examQuestion.Question_subject = (string)row["question_subject"];
        examQuestion.Question_keys = (string)row["question_keys"];
        examQuestion.A =Convert.ToString(row["a"]);
        examQuestion.B =Convert.ToString(row["b"]);
        examQuestion.C =Convert.ToString(row["c"]);
        examQuestion.D =Convert.ToString(row["d"]);
        examQuestion.examAllSubject =AllSubjectService.GetSubjectById((int)
        row["subject_id"]);            //外键
        examQuestion.examQuestionType =QuestionTypeService.GetQuestionType
        ById((int)row["question_type"]);
                                       //外键
        questionList.Add(examQuestion);
    }
    return questionList;
    }
}
```

上述数据访问层的这两个方法在知识准备里面已有说明，在此不再赘述。



3. 业务逻辑层添加的方法

在业务逻辑层的 AllQuestionsManager.cs 类中,添加方法如下:

```
public static IList<examAllQuestion> GetAllQuestionBySql(string orderType)
{
    return AllQuestionsService.GetAllQuestionBySql(orderType);
}
```

4. 用户界面表示层添加的方法

在数据访问层和业务逻辑层添加完相应的方法之后,需要在用户界面表示层作数据绑定,这样就可以实现按传递的参数获得排序的结果了。用户界面在表示层的后台 QuestionLists.aspx.cs 中编写如下两个方法。

```
protected void btnType_Click(object sender, EventArgs e)
{
    ViewState["OrderType"]="question_type";
    Pager=0;
    this.btnType.Enabled=false;
    this.btnSubject.Enabled=true;
    DataBind();
}
protected void btnSubject_Click(object sender, EventArgs e)
{
    ViewState["OrderType"]="subject_id";
    Pager=0;
    this.btnType.Enabled=true;
    this.btnSubject.Enabled=false;
    DataBind();
}
```

练　　习

1. 单项选择题

(1) DataList 的换行符模板是________。

A. SeparatorTemplate　　　B. ItemTemplate

C. TemplateField　　　D. AlternatingTemplate

(2) 下列关于 DataList 的说法中,正确的是________。

A. DataList 不会自动生成任何代码

B. DataList 不能使用 ObjectDataSource 控件进行数据绑定

C. DataList 没有内置分页功能

D. DataList 只有模板列

(3) 下列关于 PagedDataSource 的说法,正确的是________。

A. PagedDataSource 封装了数据绑定控件的分页功能





B. PagedDataSource 可以帮助我们计算总页数、当前页数、每页显示条数

C. 使用 PagedDataSource，就不能使用 ObjectDataSource

D. 使用 ObjectDataSource 可以方便地实现分页和排序

2. 简答题

(1) 分析 GridView 及 DataList 这两个控件的特点。

(2) 简述使用 PagedDataSource 分页类实现分页的思路。

实　训

实训目的

(1) 熟练掌握 DataList 控件的用法。

(2) 掌握 DataList 控件的分页和排序。

实训内容

(1) 使用 DataList 控件实现 onLineWeb 项目中管理员信息的列表显示。

(2) 对列表中的信息进行分类排序。

(3) 对列表中的信息实现翻页功能。

(4) 对列表中的信息进行删除。

实训指导

(1) 在项目中新建一页面 shixun11. aspx，使用 DataList 控件布局 shixun11. aspx 页面，并把实现功能所需的控件放置在页面上。页面布局如图 10-11 所示。



图 10-11　页面布局

(2) 按照本章介绍的方法，首先实现管理员列表显示，如图 10-12 所示。

(3) 对列表中的信息进行分类排序，根据选择的排序方式显示信息。

按管理员 ID 排序，如图 10-13 所示。

按用户名排序，如图 10-14 所示。

(4) 对列表中的信息实现翻页功能，如图 10-15 所示。



管理员列表排序方式： 管理员ID 用户名

管理员Id	用户名	密码	系统管理权限	详细	删除
1	admin	123456		详细	删除
2	teacher	123456		详细	删除
6	w	w		详细	删除

第 1 页 共 2 页 上一页 下一页

图 10-12 实现管理员列表显示

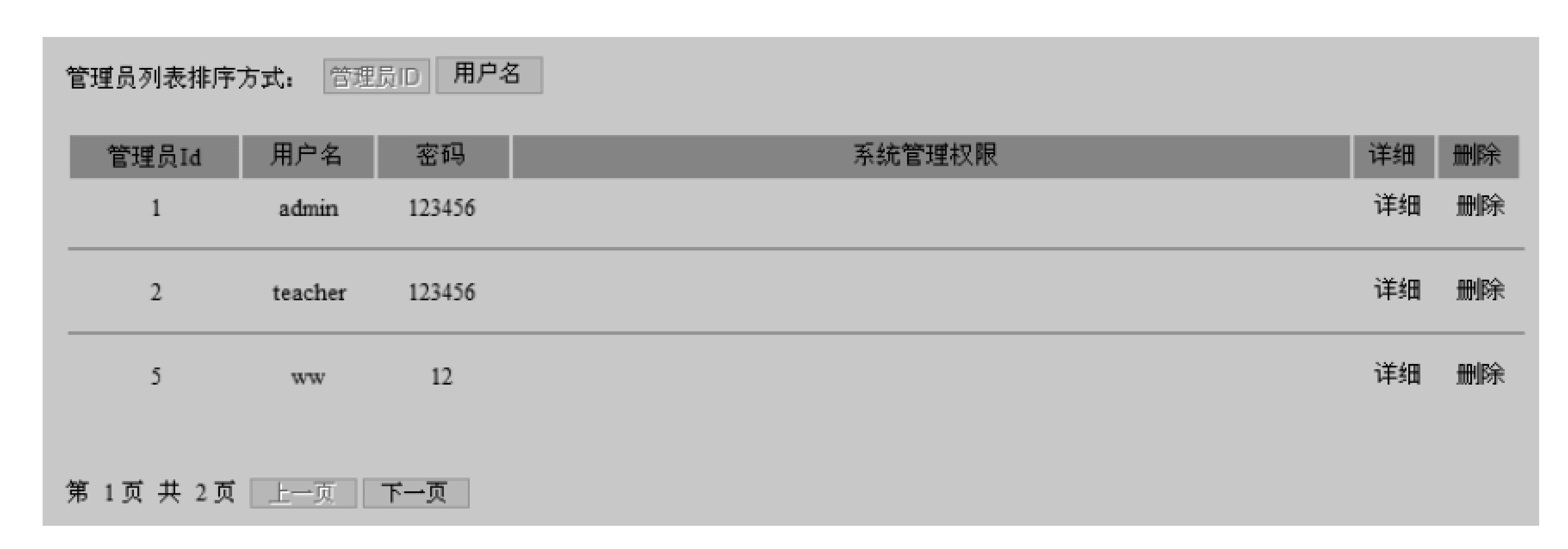


图 10-13 按管理员 ID 排序的结果



图 10-14 按用户名排序的结果



图 10-15 列表信息的翻页

（5）实现对列表中信息的删除。

在 DeleteAdminUsers.aspx 页面中，实现对任意一条信息的删除，代码如下：

```
public partial class shixunExamples_DeleteAdminUsers:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
```






```
            string adminId=Request.QueryString["qId"];
            if (adminId !=null)
            {
                Response.Write("<script>alert('确定删除此项吗?');window.location=
                'shixun11.aspx'</script>");
                  AdminManager.DeleteExamAdminById(Convert.ToInt32(adminId));
                  //Response.Redirect("QuestionLists.aspx");
            }
        }
}
```

(6) 实现上述功能的全部代码如下:

```
public partial class shixunExamples_shixun11:System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
        if (!IsPostBack)
        {
            //首次加载,赋初值
            ViewState["Page"]=0;
            ViewState["OrderType"]="";
            DataBind();
        }
    }
    private void DataBind()
    {
        PagedDataSource pdsAdminUsers=new PagedDataSource();
        //对 PagedDataSource 对象的相关属性赋值
            pdsAdminUsers.DataSource = AdminManager. GetAllAdminUsersBySql ((string)
            ViewState["OrderType"]);
        pdsAdminUsers.AllowPaging=true;
        pdsAdminUsers.PageSize=3;
        pdsAdminUsers.CurrentPageIndex=Pager;
        lblCurrentPage.Text="第 "+(pdsAdminUsers.CurrentPageIndex+1).ToString()+" 页 共 "
        +pdsAdminUsers.PageCount.ToString()+" 页";
        SetEnable(pdsAdminUsers);

        //把 PagedDataSource 对象赋给 DataList 控件
        dlAdminUsers.DataSource=pdsAdminUsers;
        dlAdminUsers.DataBind();
    }
        protected void btnAdminID_Click(object sender,EventArgs e)
    {
        ViewState["OrderType"]="admin_id";
        Pager=0;
        this.btnAdminID.Enabled=false;
        this.btnAdminName.Enabled=true;
        DataBind();
    }
```



```
    protected void btnAdminName_Click(object sender,EventArgs e)
    {
        ViewState["OrderType"]="admin_name";
        Pager=0;
        this.btnAdminID.Enabled=true;
        this.btnAdminName.Enabled=false;
        DataBind();
    }
    private void SetEnable(PagedDataSource pds)
    {
        btnPrev.Enabled=true;
        btnNext.Enabled=true;
        if (pds.IsFirstPage)
          btnPrev.Enabled=false;
        if (pds.IsLastPage)
          btnNext.Enabled=false;
    }
    protected void btnPrev_Click(object sender,EventArgs e)
    {
        Pager--;
        DataBind();
    }
    protected void btnNext_Click(object sender,EventArgs e)
    {
        Pager++;
        DataBind();
    }
    private int Pager
    {
        get
        {
            return (int) ViewState["Page"];
        }
      set
      {
          ViewState["Page"]=value;
      }
    }
}
```



任务11 第三方控件与试题的信息添加

技能目标

掌握第三方控件 FreeTextBox 的使用，能对考生试题信息进行添加。

知识目标

会使用第三方控件。

任务描述

本部分对一些第三方控件进行了介绍，同时介绍了添加试题信息的方法。

本章任务如下：

- 熟悉第三方控件 FreeTextBox 的使用方法；
- 实现对试题信息的添加。

11.1 知识准备

11.1.1 第三方控件介绍

在 Visual Studio 开发工具中，有很多控件，这些控件是由微软公司开发的常用控件。在使用过程中，又有人开发了适合自己的控件，并把它们放在网上供其他人免费使用，这些控件就是第三方控件。还有一些第三方控件是由其他软件供应商提供的，比如一些图像处理控件供应商如 LeadTools、ReadIRIS 等提供了非常专业化的图像处理的控件，且是收费的。第三方控件封装合理、功能强大，使用第三方控件的优点是可以加快开发速度，提高开发效率；缺点是如果不是开源的，需要花钱，还有就是第三方控件很可能经常升级，需要注意更新的内容。简单地说，第三方控件就是非.NET 系统中的控件。

11.1.2 FreeTextBox 控件

1. FreeTextBox 简介

FreeTextBox（简称 FTB）是一种基于 Internet Explorer 中 MSHTML 技术的 ASP.NET 开源服务器控件。这是一款优秀的自由软件（free software），我们可以轻松地将其嵌入到 WebForms 中实现 HTML 内容的在线编辑。它在新闻发布、博客写作、论坛社区等多



种 Web 系统中都有用途。FreeTextBox 能定制编辑器界面，包括 Office 2000/Office XP/Office 2003 等界面，如图 11-1 所示。

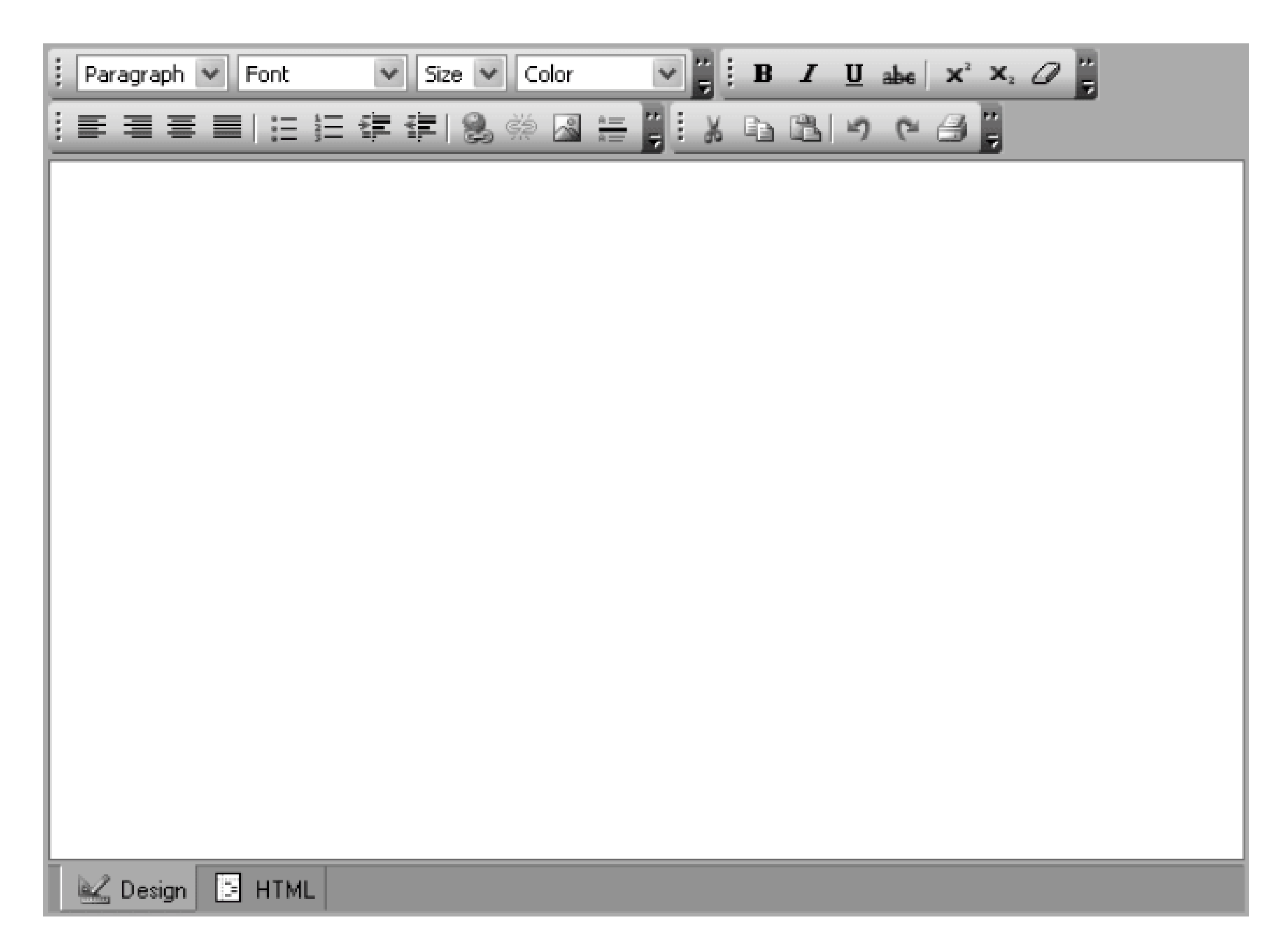


图 11-1 FreeTextBox 界面

2. FreeTextBox 的使用步骤

FreeTextBox 的使用步骤如下：

1）从网上下载 FreeTextBox

2）安装 FreeTextBox

（1）在站点目录下创建 bin 文件夹，把 FreeTextBox.dll 文件放到站点目录的 bin 文件夹中。

（2）在工具箱面板上右击，弹出快捷菜单如图 11-2 所示，选择“选择项”命令。

列表视图(L)
全部显示(S)
选择项(I)...
按字母顺序排序(O)
重置工具箱(E)
添加选项卡(A)
删除选项卡(D)
重命名选项卡(R)
上移(U)
下移(W)

图 11-2 选择“选择项”命令

（3）打开“选择工具箱项”对话框，如图 11-3 所示。单击“浏览”按钮，在出现的“打开”对话框中选择 bin 文件夹中的 FreeTextBox.dll 文件，如图 11-4 所示，之后单击“打开”按钮。

（4）安装之后工具箱中出现 FreeTextBox 选项，如图 11-5 所示。

3）使用 FreeTextBox

安装之后使用 FreeTextBox 时就像一般控件一样，从工具箱中拖到页面上即可。在 aspx 页面中生成对应的代码如下：

```
<%@ Register Assembly="FreeTextBox"Namespace=
"FreeTextBoxControls"TagPrefix="FTB"%>
  <ftb:freetextbox id="FreeTextBox1" runat= "server" Text="abc"
ToolBarStyleConfiguration="NotSet"></ftb:freetextbox>
```

从代码中可以看出 FTB 控件的命名空间是 FreeTextBoxControls。

3. FreeTextBox 的常用属性

FreeTextBox 常用属性如下。



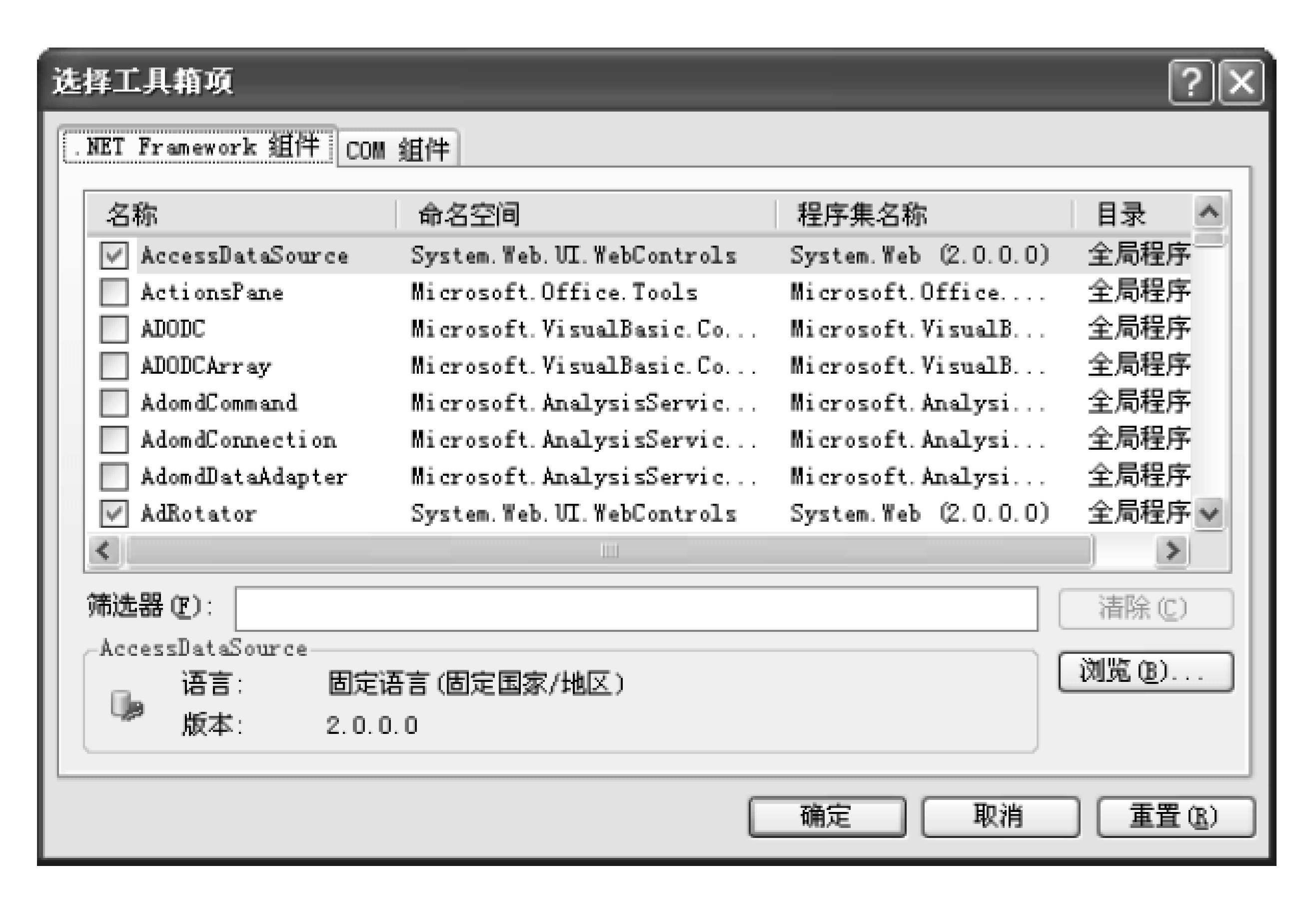


图 11-3　选择工具箱项

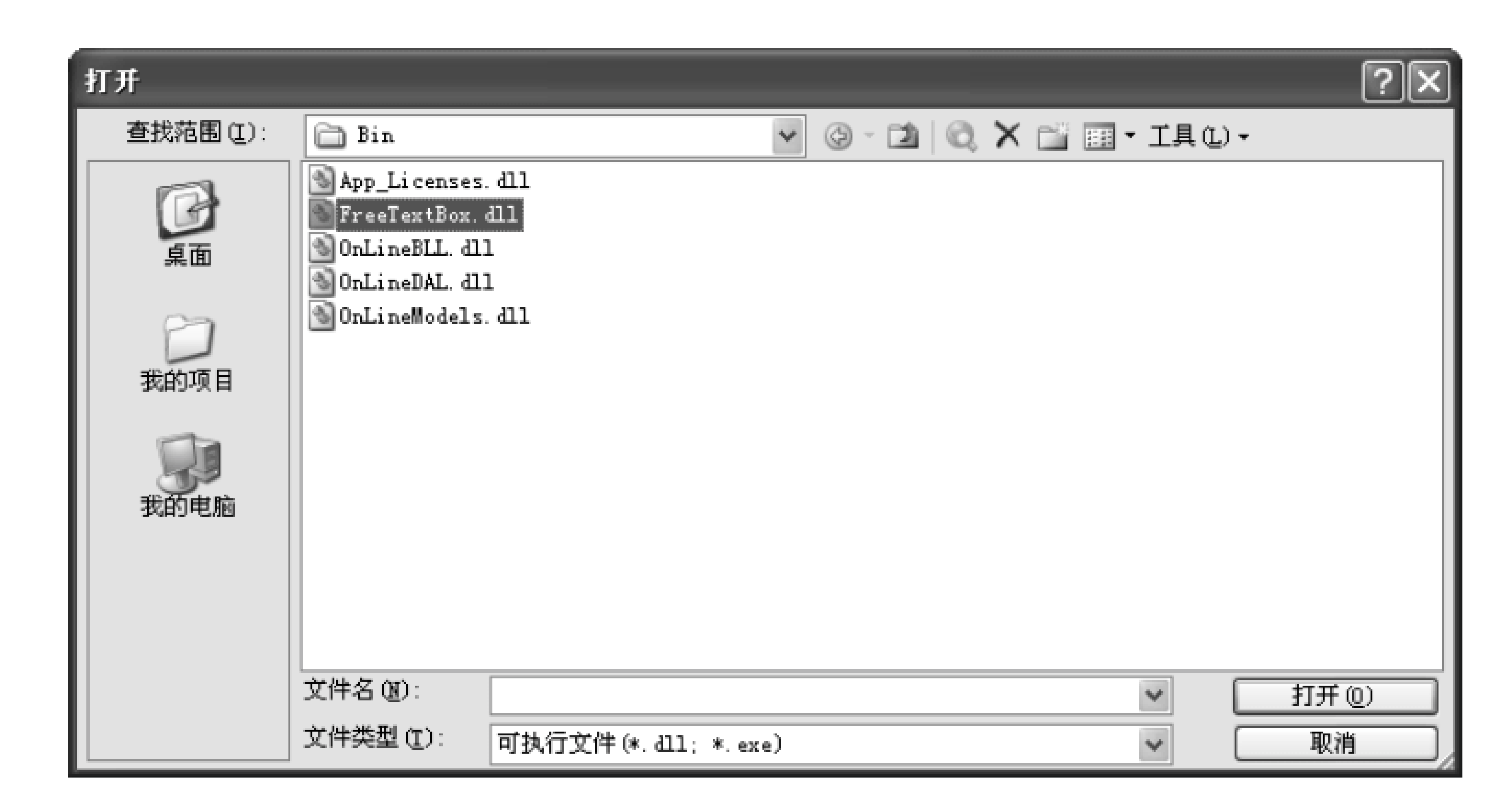


图 11-4　选择 FreeTextBox.dll 文件

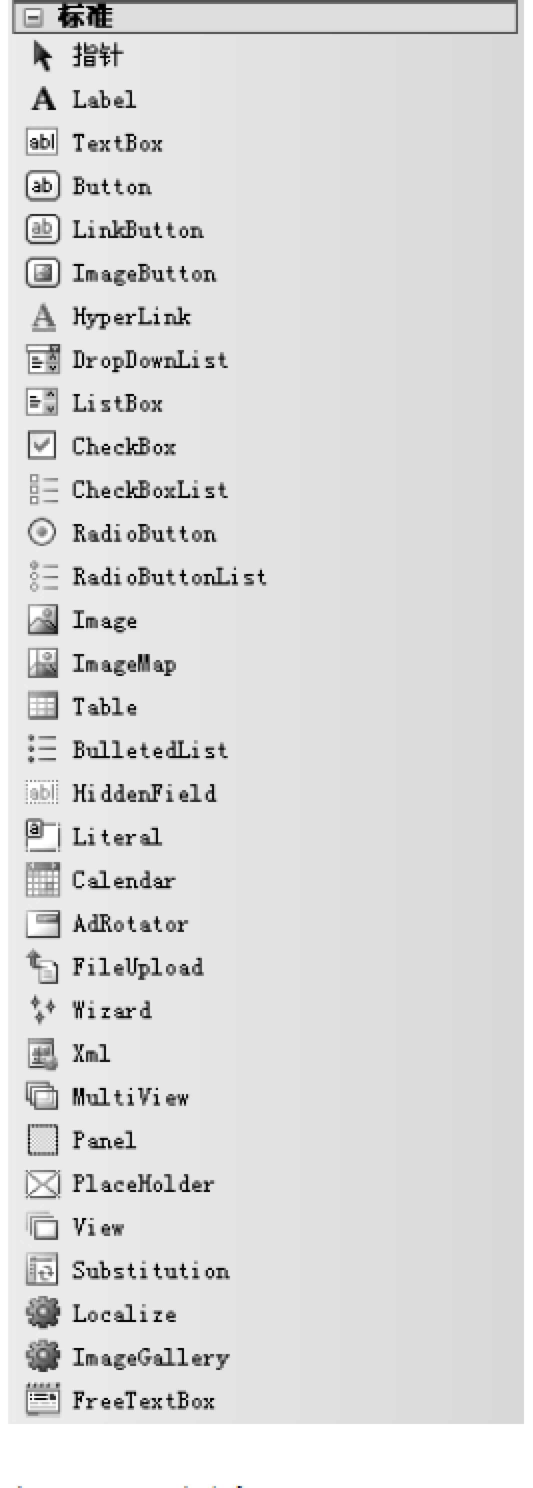


图 11-5　添加 FreeTextBox 后的工具箱

- ID：FTB 控件的 ID 号。
- Language：设置 FTB 的语言，输入"zh-CN"显示简体中文。
- Height：设置 FreeTextBox 的高度。
- Width：FTB 控件的宽。
- Text：FTB 中输入的文本内容，这是带 HTML 标记的。
- HtmlStrippedText：FTB 中输入的文本内容，这个是将 HTML 标记去掉的文本。
- ToolBarStyleConfiguration：设置 FTB 的显示风格，如图 11-6 所示。

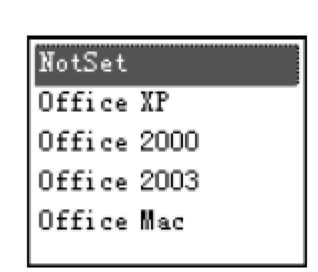


图 11-6　FTB 的显示方式

11.2　任务实施

在在线考试系统中，选择"后台管理"菜单中的"添加试题"选项，如图 11-7 所示，将打开 QuestionDetails.aspx 页面实现添加和编辑功能，如图 11-8 所示。在添加试题时用到了



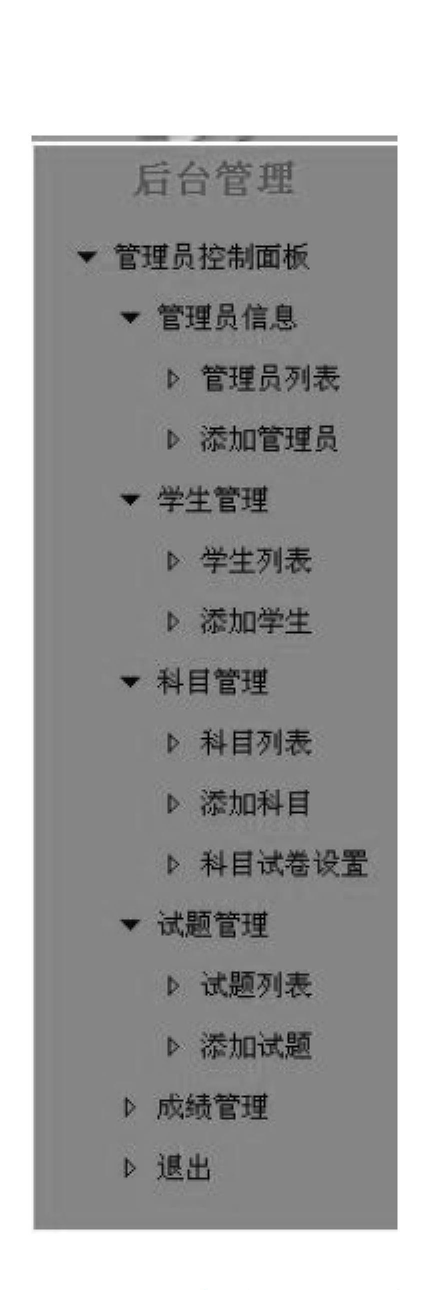


图 11-7　在线考试系统“后台管理”菜单

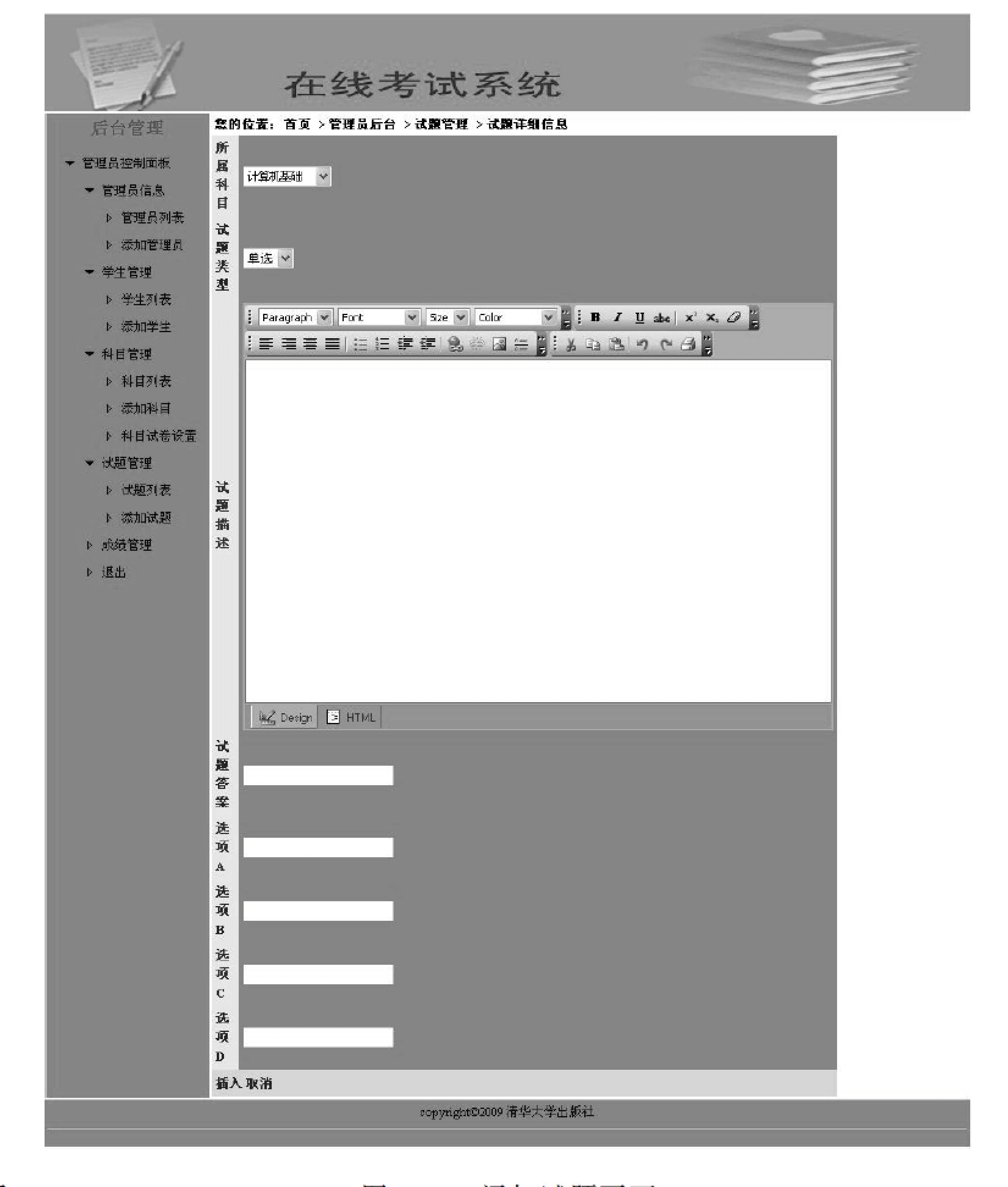


图 11-8　添加试题页面

FreeTextBox 控件。

下面就来完成添加试题功能。

1. 添加 QuestionDetails.aspx 页面

右击 admin 文件夹，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，如图 11-9 所示。在“名称”文本框中输入“QuestionDetails.aspx”，选中“选择母版页”复选框，单击“添加”按钮，打开“选择母版页”对话框，如图 11-10 所示。选择 admin 文件夹下的 AdminMaster.master 母版页，单击“确定”按钮，这样 QuestionDetails.aspx 页面就添加到项目中。

2. 在 QuestionDetails.aspx 页面中添加数据源

要添加的 ObjectDataSource 控件有 3 个。

(1) 第一个 ObjectDataSource 控件用来显示修改和添加试题信息，其主要属性设置如表 11-1 所示。





图 11-9　添加新项



图 11-10　选择母版页

表 11-1　odsQuestionDetail 控件属性设置

属　性	属　性　值	属　性	属　性　值
ID	odsQuestionDetail	UpdateMethod	ModifyExamQuestion
InsertMethod	AddQuestion	TypeName	OnLineBll. AllSubjectManager
SelectMethod	GetExamQuestionById		

除了以上参数外，还有如下重要参数。

✍ InsertParameters

在 InsertParameters 属性中添加如下参数，名称分别为：subject_id、question_diff、type_id、question_score、question_subject、question_keys、a、b、c、d，如图 11-11 所示。

✍ SelectParameters

在 SelectParameters 属性中添加参数名称为 Question_id，参数源设置为 QueryString，QueryStringField 设置为 qId，如图 11-12 所示。



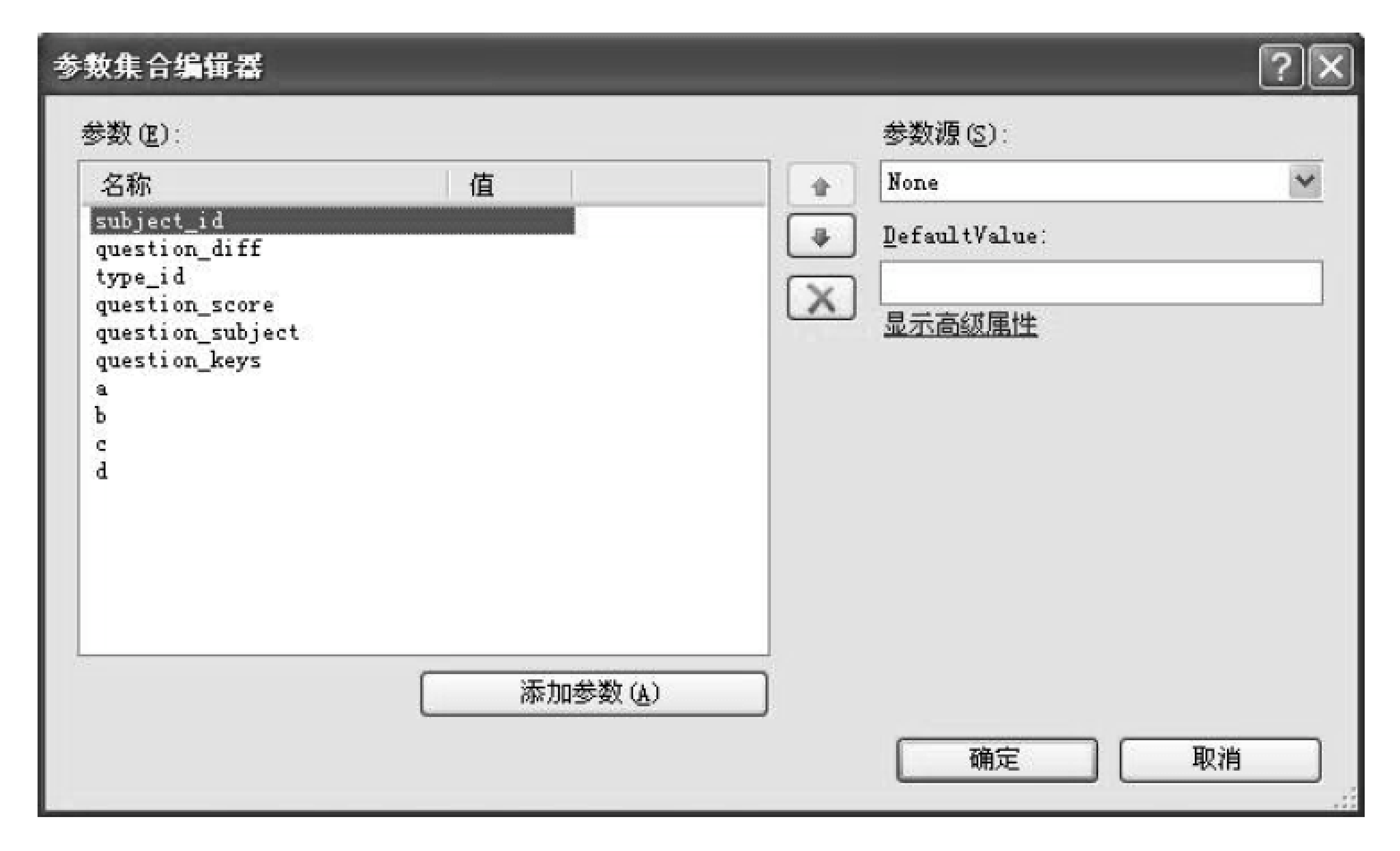


图 11-11　InsertParameters 设置

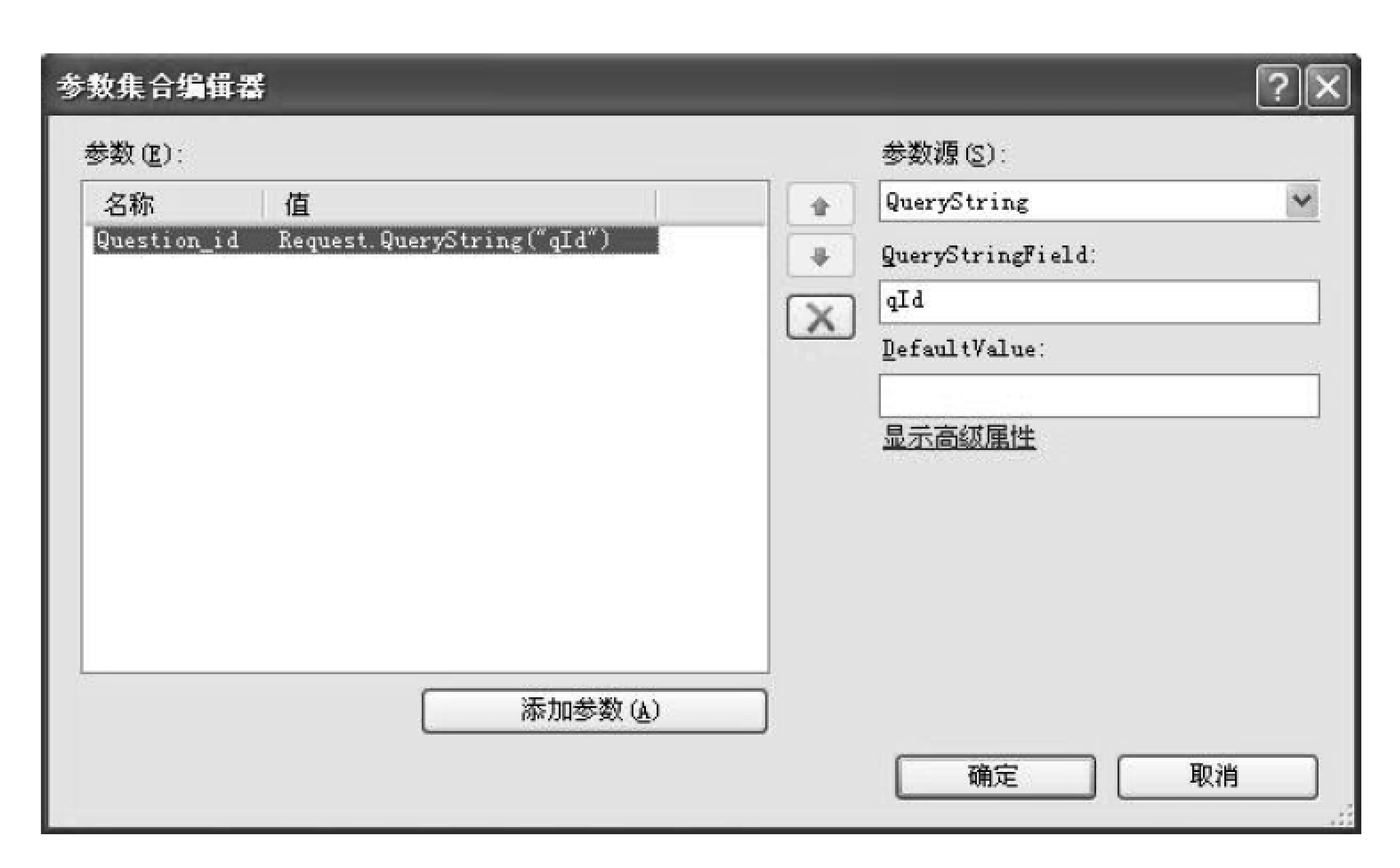


图 11-12　SelectParameters 设置

✍ UpdateParameters

在 UpdateParameters 属性中添加如下参数，名称分别为：subject_id、question_diff、type_id、question_score、question_subject、question_keys、a、b、c、d 和 Question_id，参数源设置为 QueryString，QueryStringField 设置为 qId，如图 11-13 所示。

（2）第二个 ObjectDataSource 控件用来获得所有考试科目，其主要属性设置如表 11-2 所示。

表 11-2　odsAllSubject 控件属性设置

属　性	属　性　值	属　性	属　性　值
ID	odsAllSubject	TypeName	OnLineBell.AllSubjectManager
SelectMethod	GetAllSubject		

（3）第三个 ObjectDataSource 控件用来获得所有试题类型，其主要属性设置如表 11-3 所示。



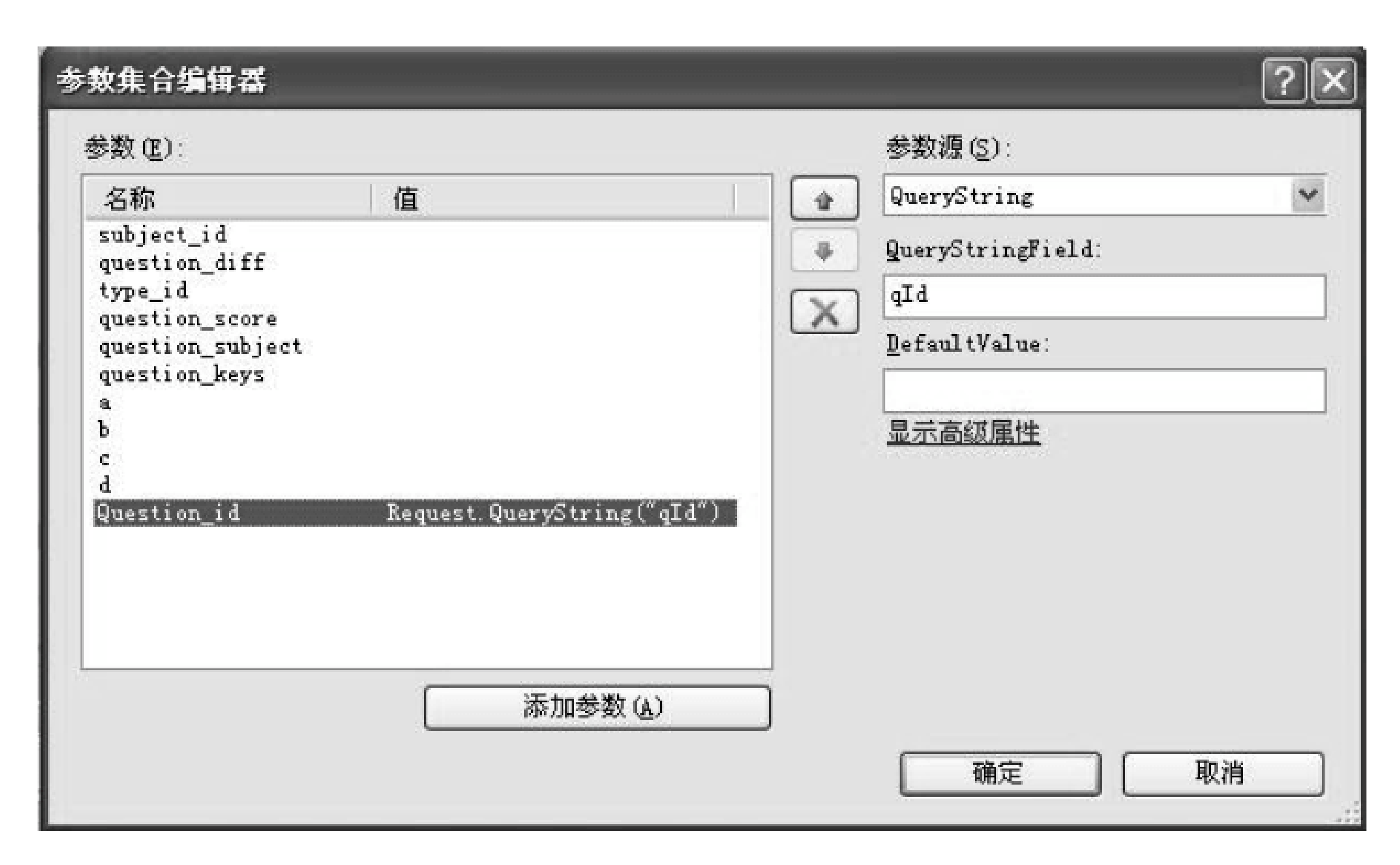


图 11-13　UpdateParameters 设置

表 11-3　odsQuestionType 控件属性设置

属　性	属　性　值	属　性	属　性　值
ID	odsQuestionType	TypeName	OnLineBell. QuestionTypeManager
SelectMethod	GetAllQuestionType		

3. 在 QuestionDetails. aspx 页面中添加 DetailsView 控件

在页面中添加一个 DetailsView 控件，用来显示和编辑试题信息，其主要属性设置如表 11-4 所示。

除了表 11-4 中的属性外，还有一个重要的属性——Fields 属性。

在属性面板上单击 Fields 属性的 Collection 后的按钮...，如图 11-14 所示，打开添加字段对话框，如图 11-15 所示。

表 11-4　dvQuestion 控件属性设置

属　性	属　性　值
ID	dvQuestion
AutoGenerateRows	False
DataSourceID	odsQuestionDetail


说明：显示、修改和添加试题信息。



图 11-14　设置 Fields 属性 1

在添加字段对话框中添加一个 BoundField 列，设置 HeaderText 属性为“试题 ID”，ReadOnly 属性设置为“true”，SortExpression 属性设置为“Question_id”，Visible 属性设置为“false”，DataField 属性设置为“Question_id”，如图 11-16 所示。

用同样的方法再添加 8 个模板列，分别设置其 HeaderText 属性为所属科目、试题类型、试题描述、试题答案、选项 A、选项 B、选项 C 和选项 D，分别设置其 SortExpression 属性为 examAllSubject、examQuestionType、Question_Subject、Question_keys、A、B、C 和 D。



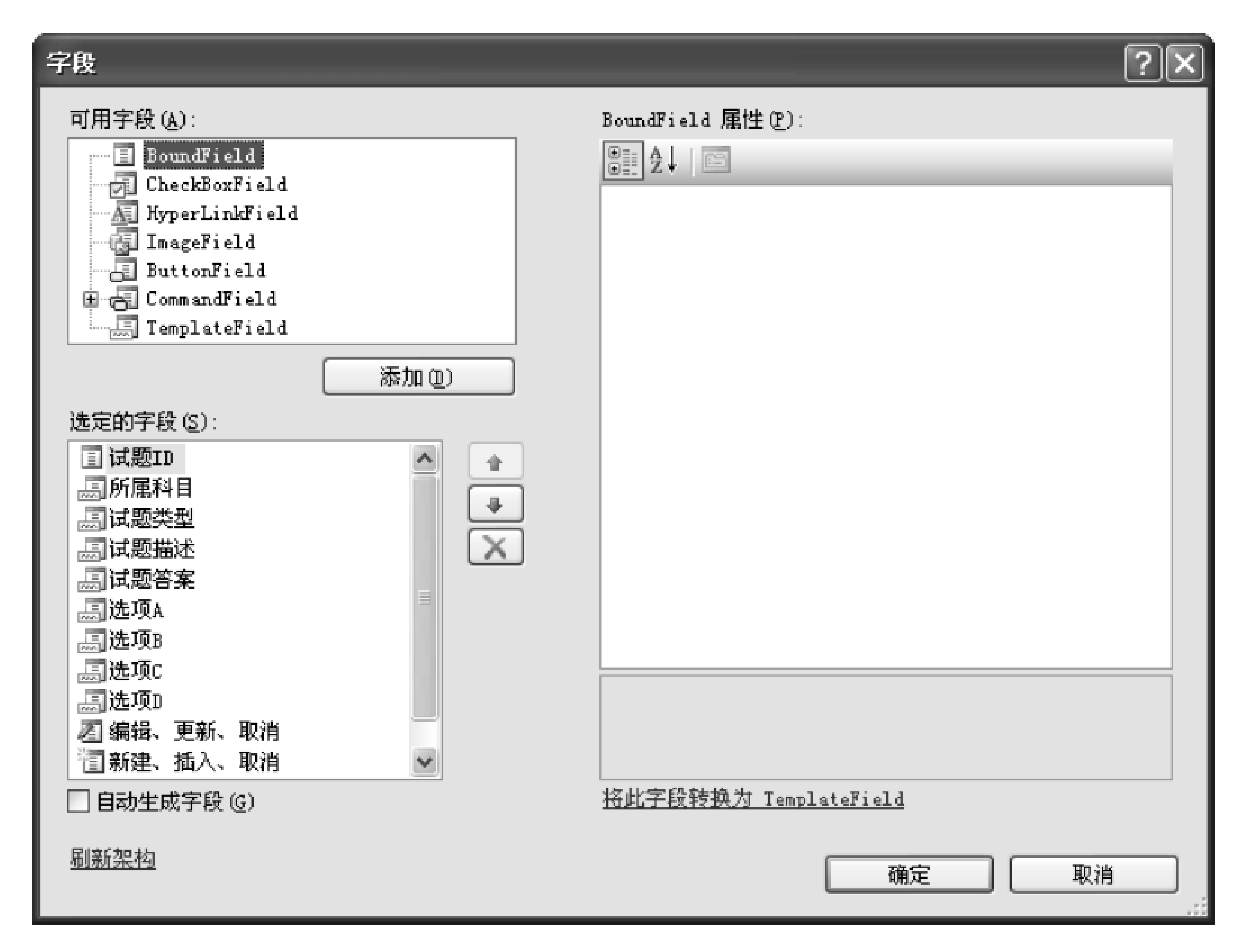


图 11-15　设置 Fields 属性 2



图 11-16　设置 Fields 属性 3

添加两个 CommandField 列，一个为编辑列，一个为新建列，如图 11-17 所示。

4. 编辑 DetailsView 控件模板列

右击 DetailsView 控件，在弹出的快捷菜单中选择“编辑模板”|“所属科目”命令，如图 11-18 所示。在弹出的窗口中添加控件，如图 11-19 所示。





图 11-17　设置 Fields 属性 4



图 11-18　编辑模板

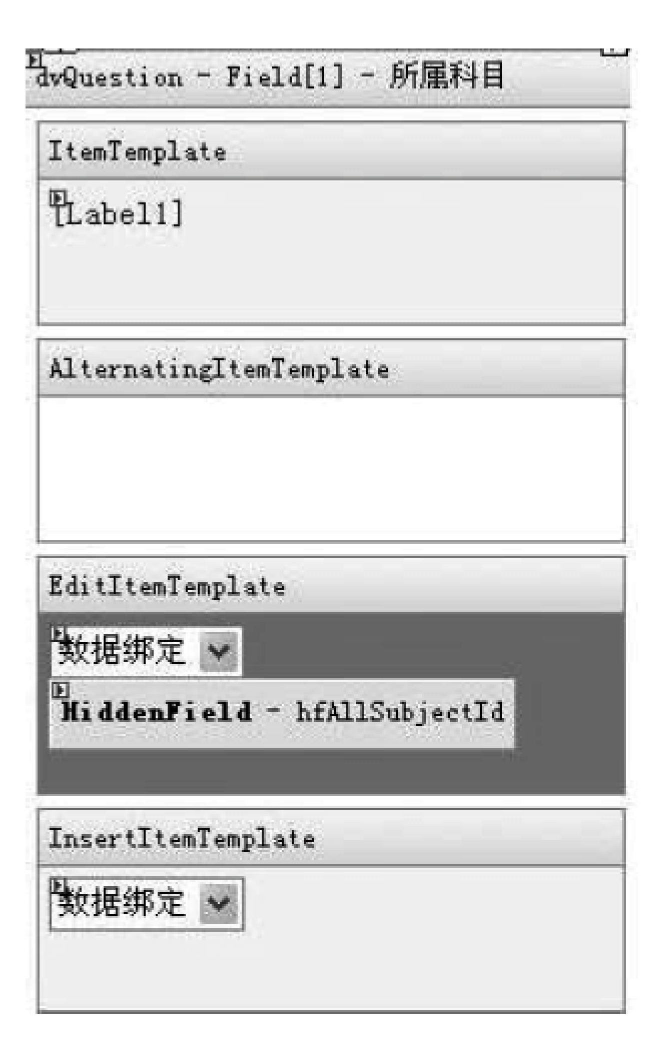


图 11-19　所属科目设置

在 EditItemTemplate 和 InsertItemTemplate 区域添加 DropDownList 控件，其属性设置如表 11-5 所示。在 ItemTemplate 区域添加一个 Label 控件，在源视图模式下添加 Text 属性的值，代码如下：

```
<asp:Label ID="Label1"runat="server"Text=
'<%# Eval("examAllSubject.Subject_name")%>'Width="61px"></asp:Label>
```

在 EditItemTemplate 区域添加一个 HiddenField 控件，其 ID 属性设置为 hfAllSubjectId，在源视图模式下添加 Value 属性值代码如下：

```
<asp:HiddenField ID="hfAllSubjectId"runat="server"Value='<%# Eval("examAllSubject.Subjectt_
id")%>'/> 。
```



表 11-5　ddlAllSubject 控件属性设置

属　　性	属　性　值	属　　性	属　性　值
ID	ddlAllSubject	DataTextField	Subject_name
DataSourceID	odsAllSubject	DataValueField	Subjectt_id

用与所属科目相同的方法编辑试题类型，如图 11-20 所示。在 EditItemTemplate 和 InsertItemTemplate 区域添加 DropDownList 控件，其属性设置如表 11-6 所示。在 ItemTemplate 区域添加一个 Label 控件，在源视图模式下添加 Text 属性的值，代码如下：

```
<asp:Label ID="Label2" runat="server"Text='<%# Eval("examQuestionType.Type_name")%>'></
asp:Label>
```

表 11-6　ddlQuestionType 控件属性设置

属　　性	属　性　值	属　　性	属　性　值
ID	ddlQuestionType	DataTextField	Type_name
DataSourceID	odsQuestionType	DataValueField	Type_id

在 EditItemTemplate 区域添加一个 HiddenField 控件，其 ID 属性设置为 hfQuestionTypeId，在源视图模式下添加 Value 属性值代码如下：

```
<asp:HiddenField ID="hfQuestionTypeId"runat="server"Value='<%# Eval("examQuestionType.Type_
id")%>'/>
```

用同样的方法编辑试题描述，如图 11-21 所示。在 EditItemTemplate 和 InsertItemTemplate 区域添加 FreeTextBox 控件，其 ID 属性设置为 ftbQuestionSubject，在源视图模式下添加 Text 属性值代码如下：

```
<FTB:FreeTextBox ID="ftbQuestionSubject" runat="server" Text='<%# Bind("Question_
subject")%>'></FTB:FreeTextBox>
```



图 11-20　试题类型设置

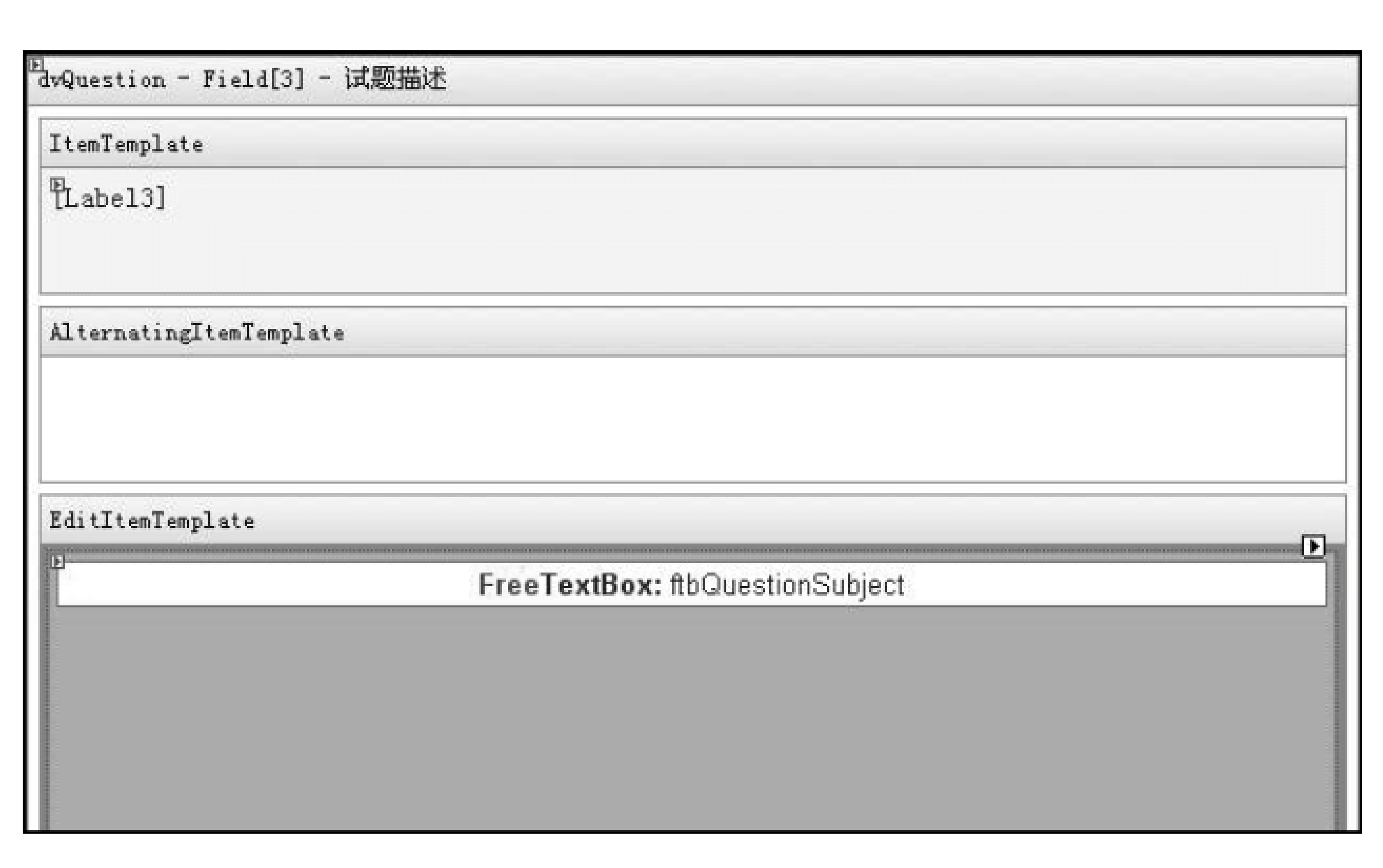


图 11-21　试题描述设置

在 ItemTemplate 区域添加一个 Label 控件，在源视图模式下添加 Text 属性的值，代码如下：



```
<asp:Label ID="Label3" runat="server"Text='<%# Bind("Question_subject")%>'></asp:
Label>
```

编辑试题答案，如图 11-22 所示。在 EditItemTemplate 和 InsertItemTemplate 区域添加 TextBox 控件，其 ID 属性设置为 txtKeys，在源视图模式下添加 Text 属性值代码如下：

```
<asp:TextBox ID="txtKeys"runat="server" Text='<%# Bind("Question_keys")%>'></asp:TextBox>
```

添加 RequiredFieldValidator 控件，其属性设置如表 11-7 所示。在 ItemTemplate 区域添加一个 Label 控件，在源视图模式下添加 Text 属性的值，代码如下：

```
<asp:Label ID="Label6"runat="server" Text='<%# Bind("Question_keys")%>'></asp:Label>
```

表 11-7　rfvKeys 控件属性设置

属　性	属 性 值	属　性	属 性 值
ID	rfvKeys	ForeColor	Red
ControlToValidate	txtKeys	ErrorMessage	试题答案不能为空

编辑选项 A，如图 11-23 所示。在 EditItemTemplate 和 InsertItemTemplate 区域添加 TextBox 控件，在源视图模式下添加 Text 属性值，代码如下：

```
<asp:TextBox ID="TextBox7"runat="server" Text='<%# Bind("A")%>'></asp:TextBox>
```

图 11-22　试题答案设置

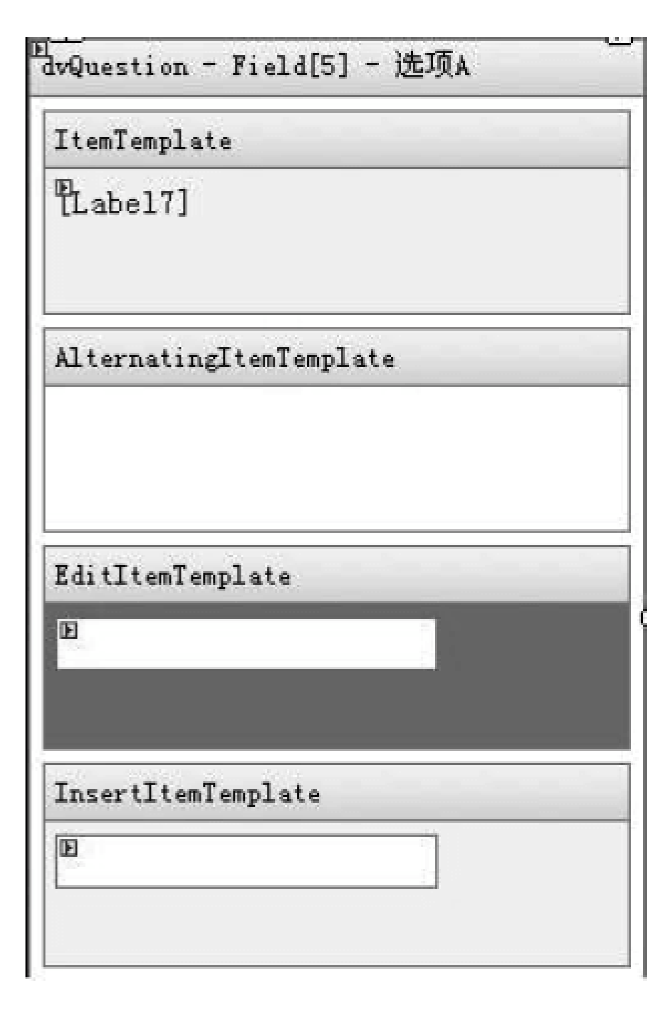


图 11-23　选项 A 设置

在 ItemTemplate 区域添加一个 Label 控件，在源视图模式下添加 Text 属性的值，代码如下：

```
<asp:Label ID="Label7" runat="server"Text='<%# Bind("A")%>'></asp:Label>
```

选项 B、选项 C 和选项 D 的编辑与选项 A 类似，只不过 Text 属性绑定的分别是 B、C 和 D，这里不再赘述。



5. 功能实现

该页面的主要功能实现代码如下：

```
protected void Page_Load(object sender,EventArgs e)
{
    if (Request.Params["qId"] ==null)
    {
        this.dvQuestion.DefaultMode =DetailsViewMode.Insert;

    }
}
//更新试题
protected void dvQuestion_ItemUpdating(object sender,DetailsViewUpdateEventArgs e)
{
    //获得试题科目下拉列表的值
     DropDownList ddlAllSubject = this. dvQuestion. FindControl ( " ddlAllSubject ") as
     DropDownList;
    //获得试题类型下拉列表的值
     DropDownList ddlQuestionType = this. dvQuestion. FindControl ( " ddlQuestionType ") as
     DropDownList;
    //添加试题科目 ID 的参数
    this.odsQuestionDetail.UpdateParameters.Add("Subjectt_id",ddlAllSubject.Selected
    Value);
    //添加试题类型 ID 的参数
    this.odsQuestionDetail.UpdateParameters.Add("Type_id",ddlQuestionType.SelectedValue);
}
protected void dvQuestion_DataBound(object sender,EventArgs e)
{
    //编辑试题时,通过隐藏框获得源试题的相关信息
    if (this.dvQuestion.CurrentMode ==DetailsViewMode.Edit)
    {
        //获得试题科目下拉列表的值
           DropDownList ddlAllSubject = this. dvQuestion. FindControl ( " ddlAllSubject ") as
           DropDownList;
        //获得试题类型下拉列表的值
        DropDownList ddlQuestionType =this.dvQuestion.FindControl("ddlQuestion
        Type") as DropDownList;
        //通过隐藏域获得源信息
        HiddenField hfAllSubjectId =this.dvQuestion.FindControl("hfAll
        SubjectId") as HiddenField;
        HiddenField hfQuestionTypeId =this.dvQuestion.FindControl
        ("hfQuestionTypeId") as HiddenField;
        //将原来信息绑定到 DropDownList
        ddlAllSubject.SelectedValue =hfAllSubjectId.Value.Trim();
        ddlQuestionType.SelectedValue =hfQuestionTypeId.Value.Trim();
    }
}
protected void dvQuestion_ItemInserted(object sender,DetailsViewInsertedEvent
Args e)
{
    //添加成功后跳转到试题列表页面
    Response.Redirect("ListAllQuestions.aspx");
```



```
}
    protected void dvQuestion_ItemInserting(object sender,
    DetailsViewInsertEventArgs e)
{
    //获得试题科目下拉列表的值
      DropDownList ddlAllSubject = this. dvQuestion. FindControl ( " ddlAllSubject ") as
      DropDownList;
    //获得试题类型下拉列表的值
      DropDownList ddlQuestionType = this. dvQuestion. FindControl ( " ddlQuestionType ") as
      DropDownList;
    //将下拉列表的值绑定到数据控件上
    this.odsQuestionDetail.InsertParameters.Add("Subjectt_id",ddlAllSubject.Selected
    Value);
    this.odsQuestionDetail.InsertParameters.Add("Type_id",ddlQuestionType.Selected
    Value);
}
```

练　　习

1. 单项选择题

（1）SQL Server. NET 数据提供的程序类位于________命名空间。

A. System. Data. SqlClient

B. System. Data. SqlServer

C. System. Data. SqlCommand

D. System. Data. Sql

（2）在 ADO. NET 中，数据适配器用于在________之间交换数据。

A. 数据源和数据源　　B. 数据集和数据集

C. 数据源和数据集　　D. 数据源和数据集或数据集与数据集

（3）ASP. NET 中，为了执行返回 DataReader 对象的命令，要使用 Command 对象的________方法。

A. ExecuteReader　　B. ExecuteScalar

C. ExecuteNonQuery　　D. ExecuteQuery

（4）要在 ASP. NET 页面中使用 DataGrid 控件来绑定并显示一张表的数据，需要设置其________属性来指定数据源。

A. ID　　B. Style　　C. DataSource　　D. DataBind

2. 简答题

（1）ADO. NET 中常用的对象有哪些？分别对其进行描述。

（2）要连接 SQL Server 2000 数据库，需引用的命名空间是什么？SQL Server. NET 的 4 个核心对象是什么？



实　　训

实训目的

（1）熟练掌握对数据的各种操作。

（2）熟悉第三方控件 FreeTextBox 的使用方法。

实训内容

（1）设计新闻发布系统。

（2）使用 FreeTextBox 控件实现新闻的编辑和添加功能。

实训指导

1）后台数据库设计

创建数据库 a，创建表 news，字段设置如图 11-24 所示。



图 11-24　news 表的设计

2）前台界面设计

（1）创建网站 ftb，按上面的步骤把 FreeTextBox 控件添加到工具箱中。

（2）设计 add_news.aspx 页面，结果如图 11-25 所示。添加一个 TextBox 控件，用来输入新闻标题。添加一个 FreeTextBox 控件，用来编辑新闻内容。添加两个 Button 控件，一个用来提交新闻，一个用来取消新闻内容。

3）后台功能代码实现

实现功能的代码如下：

```
protected void Button1_Click(object sender,EventArgs e)
{
    SqlConnection con=new SqlConnection();
    con.ConnectionString="server=(local);uid=sa;pwd=sa;database=a;";
    SqlCommand cmd=new SqlCommand();
    cmd.CommandText="insert into news values('"+txttitle.Text+"','"+FreeTextBox1.Text+"')";
    cmd.Connection=con;
```







图 11-25　add_news. aspx 页面

```
    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();

}
protected void Button2_Click(object sender,EventArgs e)
{
    txttitle.Text="";
    FreeTextBox1.Text="";
}
```

设计完成后，运行效果如图 11-26 所示。

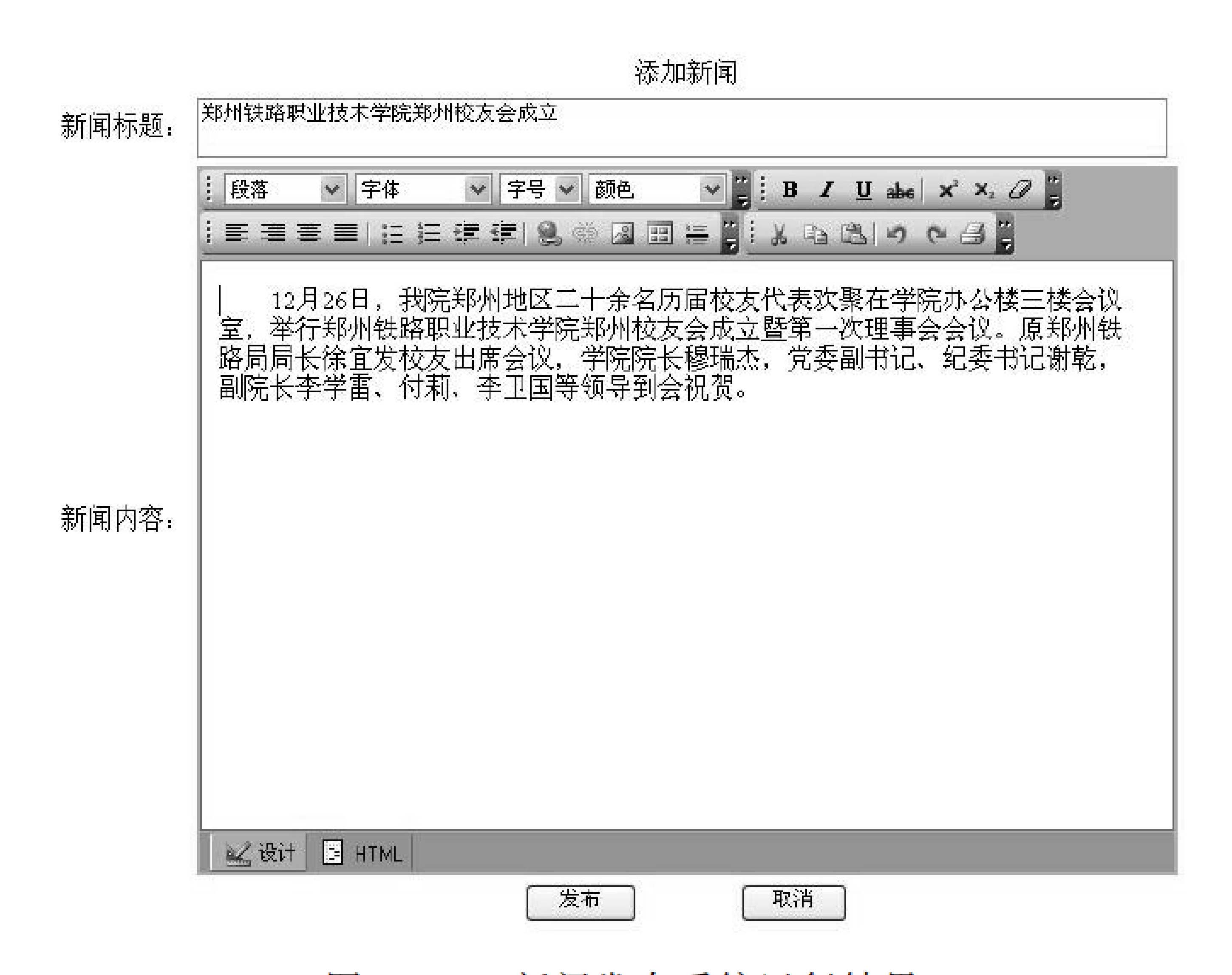


图 11-26　新闻发布系统运行结果



任务12 Repeater控件与前台试题显示

技能目标

会获取科目考试信息;能根据科目考试信息和考生选择信息,随机出现考试内容;能够统计出考生总分,保存考生试题内容。

知识目标

掌握 Repeater 控件的使用,理解单选、多选控件的使用;了解全局变量的使用。

任务描述

在线考试过程主要分三步:首先,教师在考试前需要设置科目考试信息,如试卷中都有哪些题型,每种题型多少道题,每道题多少分;其次,学生需选择考试科目,输入个人信息,系统根据科目考试信息和考生选择信息,随机出现考试内容,学生进行答题;最后,系统要能够统计出考生总分,需要把考生试题内容和答案进行备份以便以后查询。本章将来完成以上工作。

本章任务如下:

- ◆ 认识 Repeater 控件;
- ◆ 获取科目考试信息;
- ◆ 用 Repeater 控件显示试题信息;
- ◆ 统计考生成绩;
- ◆ 保存考生试题信息。

12.1 知 识 准 备

12.1.1 Repeater 控件

前面已经学习了 GridView、DetailsView 和 DataList 数据显示控件,下面将认识一种新的数据显示控件——Repeater 控件。

DataGrid 和 DataList 控件提供了各种特性,通过这些特性可以很容易地在一个 Web 页面上以列表形式显示数据。但是,如果不想使用 HTML 表格形式,就会用到 Repeater 控件了,Repeater 控件提供显示了需要数据的灵活性。

Repeater 控件本身不具有固定的外观,通过页眉模板、数据项模板、交替数据项模板、分割模板以及页脚模板,可以灵活地控制数据的显示格式。



Repeater 控件一般用于数据的显示，本身不具备编辑、分页、排序等功能（GridView 控件具有这些功能），如果需要这些功能，则需要编码来实现。

Repeater 控件不会自动生成任何用于布局的代码，使用时要通过模板来定义格式，从而灵活地控制记录的显示。Repeater 控件包含 5 个模板，如表 12-1 所示。

表 12-1　Repeater 控件的模板类型控

模　　板	说　　明
ItemTemplate	数据项模板，定义列表中项目的内容，数据源中数据的每一行都显示一次。要显示 ItemTemplate 中的数据，需要声明一个或多个 Web 服务器控件并利用 Repeater 控件的数据源中的字段设置其数据绑定表达式
AlternatingItemTemplate	交替数据项模板，与 ItemTemplate 元素类似，但在 Repeater 控件中隔行呈现一次。通过设置 AlternatingItemTemplate 元素的样式属性，可以为其指定不同的外观
SeparatorTemplate	分隔模板，在各行之间呈现的元素，通常是分行符（ 标记）、水平线（<hr> 标记）等
HeaderTemplate	页眉模板，在所有数据绑定行呈现之前呈现一次的元素。典型的用途是开始一个容器元素（如表）
FooterTemplate	页脚模板，在所有数据绑定行呈现之后呈现一次的元素。典型的用途是关闭在 HeaderTemplate 项中打开的元素（使用</table>这样的标记）

与前面学习的 GridView 控件和 DataList 控件不同，Repeater 控件不能使用视图方式添加和编辑这些模板，只能在代码中输入，并且 Repeater 控件缺少样式属性，要控制样式，也只能输入样式代码。但是 Repeater 灵活性强，主要用于一些对性能和灵活性要求比较高的数据的显示。

12.1.2　Repeater 控件使用举例

下面举例说明 Repeater 控件的使用，编写一个 ASP. NET 网页，用 Repeater 控件显示在线考试系统中的试题类型，如图 12-1 所示。具体步骤如下：

（1）启动 Visual Studio 2005，新建一网站，生成一个默认名称为“Default. aspx”的网页。

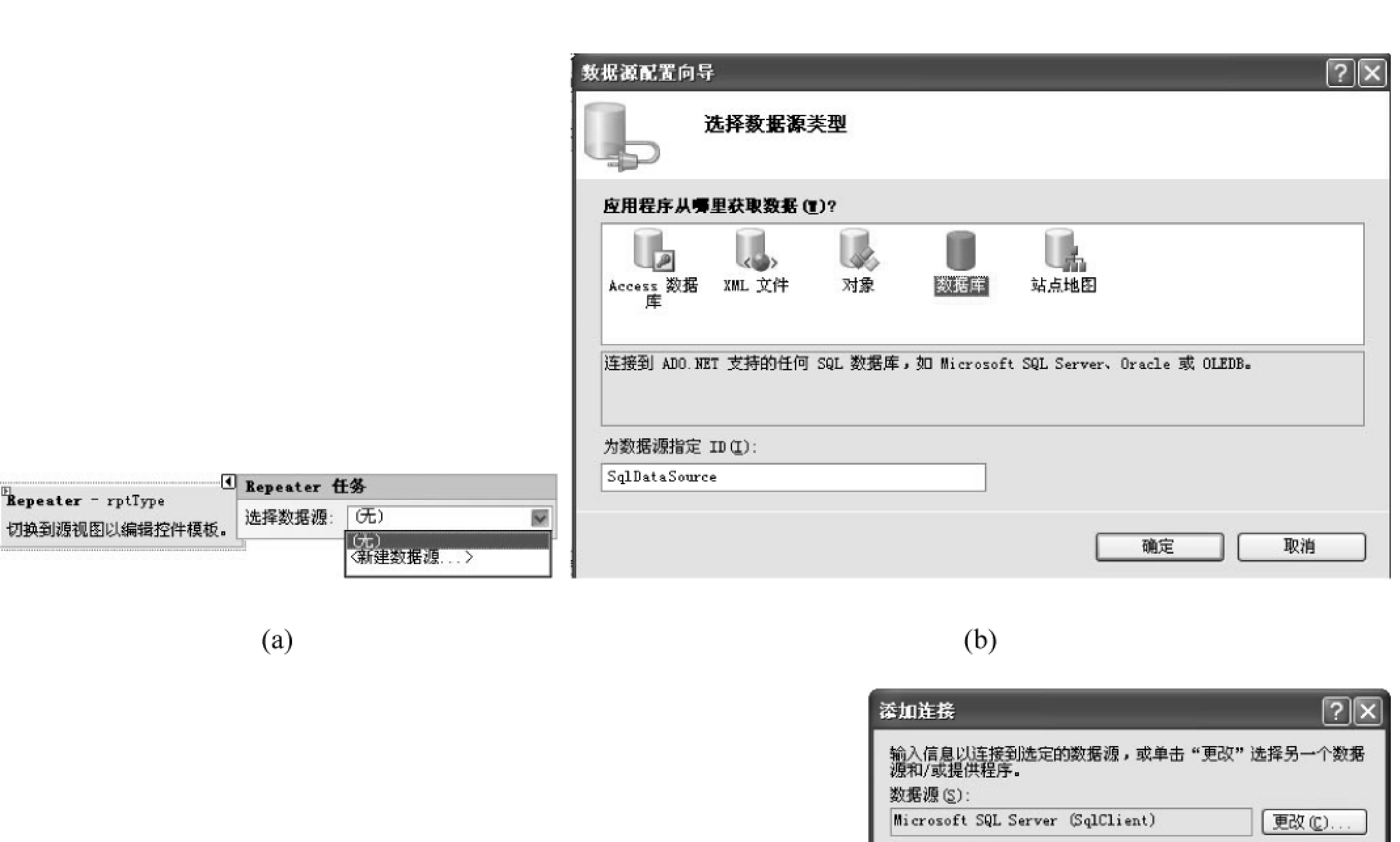
（2）从工具箱中拖曳 Repeater 控件到设计视图。选择该按钮控件，在属性窗口修改它的 ID 属性值为“rptType”。

（3）为 Repeater 控件新建数据源，可以使用设计视图，如图 12-2 所示。



图 12-1　使用 Repeater 控件显示效果





(a) (b)

配置数据源 - SqlDataSource1
选择您的数据连接
应用程序连接数据库应使用哪个数据连接(W)?
新建连接(C)...
连接字符串(S)
< 上一步(P) 下一步(N) > 完成(F) 取消



(c) (d)

配置数据源 - SqlDataSource1
选择您的数据连接
应用程序连接数据库应使用哪个数据连接(W)?
1a381975074c40c.onlineExam1.dbo
新建连接(C)...
连接字符串(S)
< 上一步(P) 下一步(N) > 完成(F) 取消

(e)

图 12-2 为 Repeater 控件配置数据源







(f)

图　12-2(续)

也可以在源视图中添加配置数据源代码,代码如下:

```
<asp:Repeater ID="rptType" runat="server" DataSourceID="SqlDataSource1" >
</asp:Repeater>
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="Data Source=.;
      Initial Catalog=onLineExam1;Integrated Security=True"
      ProviderName="System.Data.SqlClient"
      SelectCommand="SELECT[type_id],[type_name]FROM[exam_questionType]">
</asp:SqlDataSource>
```

(4) 在源视图中为 Repeater 控件添加 HeaderTemplate 模板、ItemTemplate 模板、AlternatingItemTemplate 模板 和 FooterTemplate 模板,代码如下:

```
<asp:Repeater ID="rptType"runat="server"DataSourceID="SqlDataSource1">
<HeaderTemplate>
  <ul>
</HeaderTemplate>
<ItemTemplate>
   <li>
       <asp:Label ID="lblTypeId"runat="server"Text='<%#Bind("Type_id")%>'>
       </asp:Label>
       <asp:Label ID="lblTypeName"runat="server"Text='<%#Bind("Type_name")%>'>
       </asp:Label>
   </li>
</ItemTemplate>
<AlternatingItemTemplate>
   <li style="font-weight:600">
       <asp:Label ID="lblTypeId" runat="server" Text='<%#Bind("Type_id") %>'>
       </asp:Label>
       <asp:Label ID="lblTypeName" runat="server" Text='<%#Bind("Type_name") %>'>
       </asp:Label>
  </li>
```





```
</AlternatingItemTemplate>
<FooterTemplate>
  </ul>
</FooterTemplate>
</asp:Repeater>
```

HeaderTemplate 模板中是项目符号开始标志
;FooterTemplate 模板中是项目符号结束标志;ItemTemplate 模板中是显示项,定义列表中项目的内容;AlternatingItemTemplate 模板中显示的内容和 ItemTemplate 模板相同,只是显示样式不同。AlternatingItemTemplate 模板中多了 style="color:Maroon",用来定义显示样式。

小贴士

在 Repeater 控件中,ItemTemplate 和 AlternatingItemTemplate 模板内的控件可以绑定数据,HeaderTemplate、FooterTemplate 和 SeparatorTemplate 模板中不能绑定数据。

12.2 任务实施

12.2.1 考试之前的准备工作

1. 考生登录

考生进入考试系统的第一个页面是功能选择页面,如图 12-3 所示。



图 12-3　功能选择页面

在图 12-3 中选择“开始考试”选项之后,进入登录页面,如图 12-4 所示。选择考试科目,输入考生学号,之后单击“开始考试”按钮。

当单击“开始考试”按钮时,触发事件 btnStartExam_Click,验证学号是否输入正确。如果正确,将考生学号和考试科目代码分别保存到 Session 变量 LoginStudent 和 Lession 中,



图 12-4 考生登录页面

之后转向考试开始页面(startExam. aspx)。

2. 获取科目考试信息

在 startExam. aspx 页面显示试题之前,要先根据考生选择的考试科目代码获取该科目的考试信息。考试前,任课老师在考试前已经设置了科目考试信息(试卷中有哪些题型,每种题型多少道题,每道题多少分),并将这些信息保存在 exam_papermg 数据表中,在此需要获取这些信息。

(1) 在数据访问层中添加方法 GetPaperMessageById,该方法的功能是根据科目 ID 查询该科目的试卷信息(包括试卷名称、考试试卷、单选题数量、多选题数量、判断题数量以及各类题目的分数等)。代码如下:

```
public static examPapermg GetPaperMessageBySid(int SubjectId)
{
    string sql="select * from exam_papermg WHERE subject_id=@papermgId";
    int subjectId;
    //使用 using 指令及时释放资源,调用 ConnDBHelper 类的 GetReader 方法根据 sql 语句生成
    //reader 对象
    using (SqlDataReader reader=ConnDBHelper.GetReader(sql,new SqlParameter("@
   papermgId",SubjectId)))
  {
    if (reader.Read())
    {
  //实例化一个 exampapermg 对象 papgerMessage 并根据查询到的内容给对象赋值
        examPapermg papgerMessage=new examPapermg();
        papgerMessage.Paper_id= (int)reader["paper_id"];
        papgerMessage.Paper_name= (string)reader["paper_name"];
        papgerMessage.Test_time= (int)reader["test_time"];
        papgerMessage.Type1_num= (int)reader["type1_num"];
        papgerMessage.Type2_num= (int)reader["type2_num"];
        papgerMessage.Type3_num= (int)reader["type3_num"];
        papgerMessage.Score1= (int)reader["score1"];
        papgerMessage.Score2= (int)reader["score2"];
        papgerMessage.Score3= (int)reader["score3"];
        subjectId= (int)reader["subject_id"]; //外键
        //及时关闭 reader
        reader.Close();
```



```
                //subjectId是外键,使用外键对象来表示实体之间的关系
                papgerMessage.examAllSubject=AllSubjectService.GetSubjectById(subjectId);
                return papgerMessage;
            }
            else
            {
                reader.Close();
                return null;
            }
        }
    }
```

(2) 由于用户界面表示层不能直接访问数据访问层的代码,需要在业务逻辑层添加如下代码。

```
public static examPapermg GetPaperMessageBySid(int SubjectId)
{
    return PapermgService.GetPaperMessageBySid(SubjectId);
}
```

(3) 在用户界面表示层的 startExam.cs 文件中定义一个方法 DataBind。在该方法中,首先,从 Session["Lession"]中得到科目代码的值,根据科目代码调用业务逻辑层 PapermgManager 中的 GetPaperMessageBySid 方法得到考试科目信息,从科目信息中获得试卷名称在 Label 标签 lblTitle 中显示,获得考试总时间、单选题数量、多选题数量和判断题数量分别保存在变量 intTryTime、num1、num2 和 num3 中。当页面加载时,在 Page_Load 方法中调用 DataBind 方法。代码如下:

```
public void DataBind()
{
    String subjectId=Session["Lession"].ToString();
    int sId=Convert.ToInt32(subjectId);
    examPapermg paperMessage=PapermgManager.GetPaperMessageBySid(sId);
    paperName=paperMessage.Paper_name;
    lblTitle.Text=paperName;
    intTryTime=paperMessage.Test_time;
    int num1=paperMessage.Type1_num;
    int num2=paperMessage.Type2_num;
    int num3=paperMessage.Type3_num;
}

protected void Page_Load(object sender,EventArgs e)
{
  …
    if (!Page.IsPostBack)
  {
      //首次加载,赋初值
      DataBind();
  }
}
```



12.2.2 使用 Repeater 控件显示试题信息

1. 试卷布局

如果考生选择科目和输入学号信息正确,登录后考试显示页面如图 12-5 所示。浏览器的状态栏中显示考试总时间和剩余时间;试卷内容分成三大部分,分别为标题和登录考试数信息部分、试题显示部分、提交按钮部分,各部分之间由水平线分隔;标题和按钮显示较简单,这里主要讨论试题的显示。

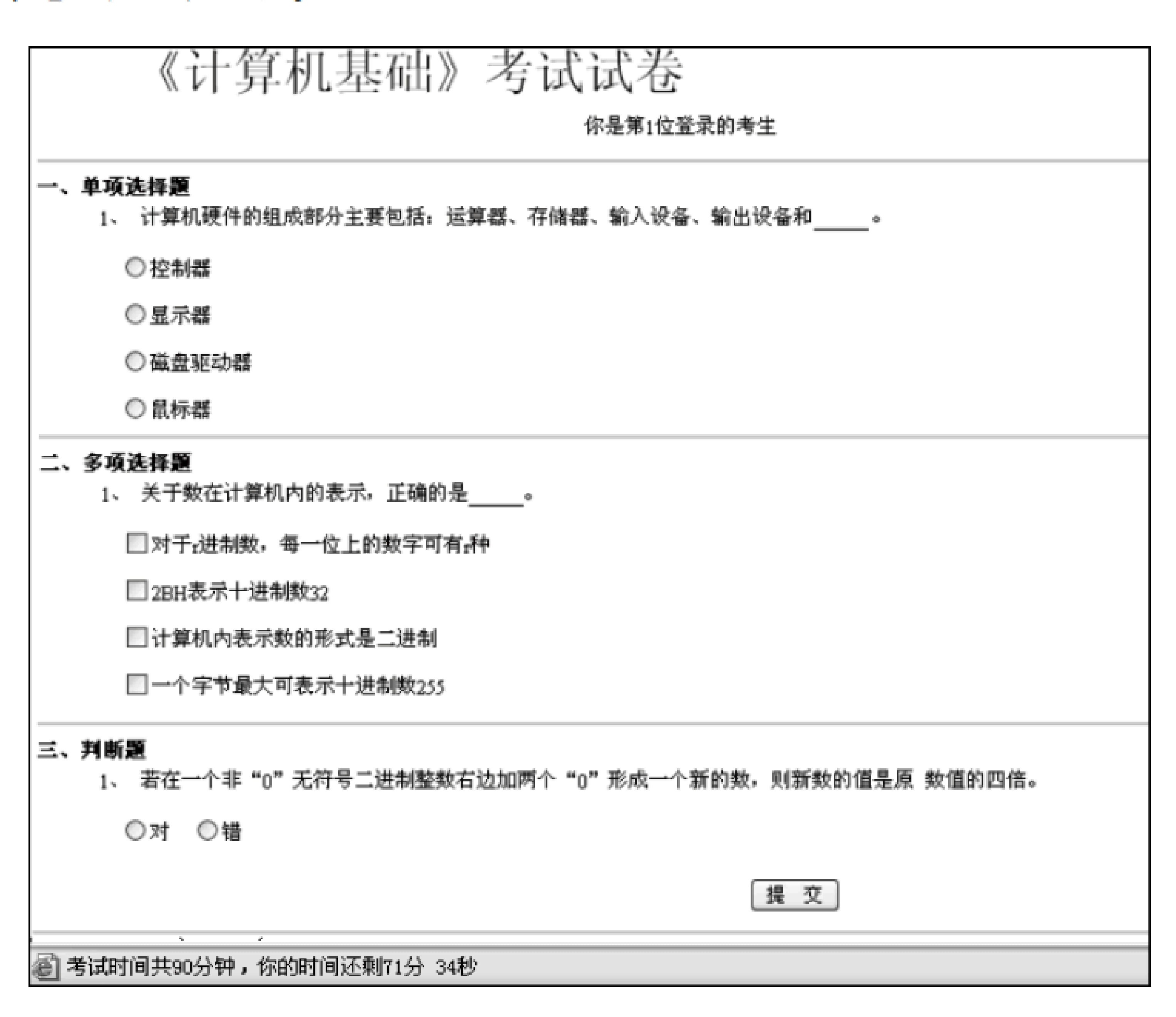


图 12-5 考试试卷布局

下面说明如何实现图 12-5 中显示的试卷。为了便于管理,考生所有试题都只在 Panel 控件中,每一种题型用一个 Repeater 控件来显示。

在设计视图中,从工具箱中拖曳一个 Panel 控件到页面中,在 Panel 控件中输入"一、单项选择题",设置字体格式,然后从工具箱拖曳一个 Repeater 控件到 Panel 控件。多项选择题和判断题的设置类似,结果如图 12-6 所示。

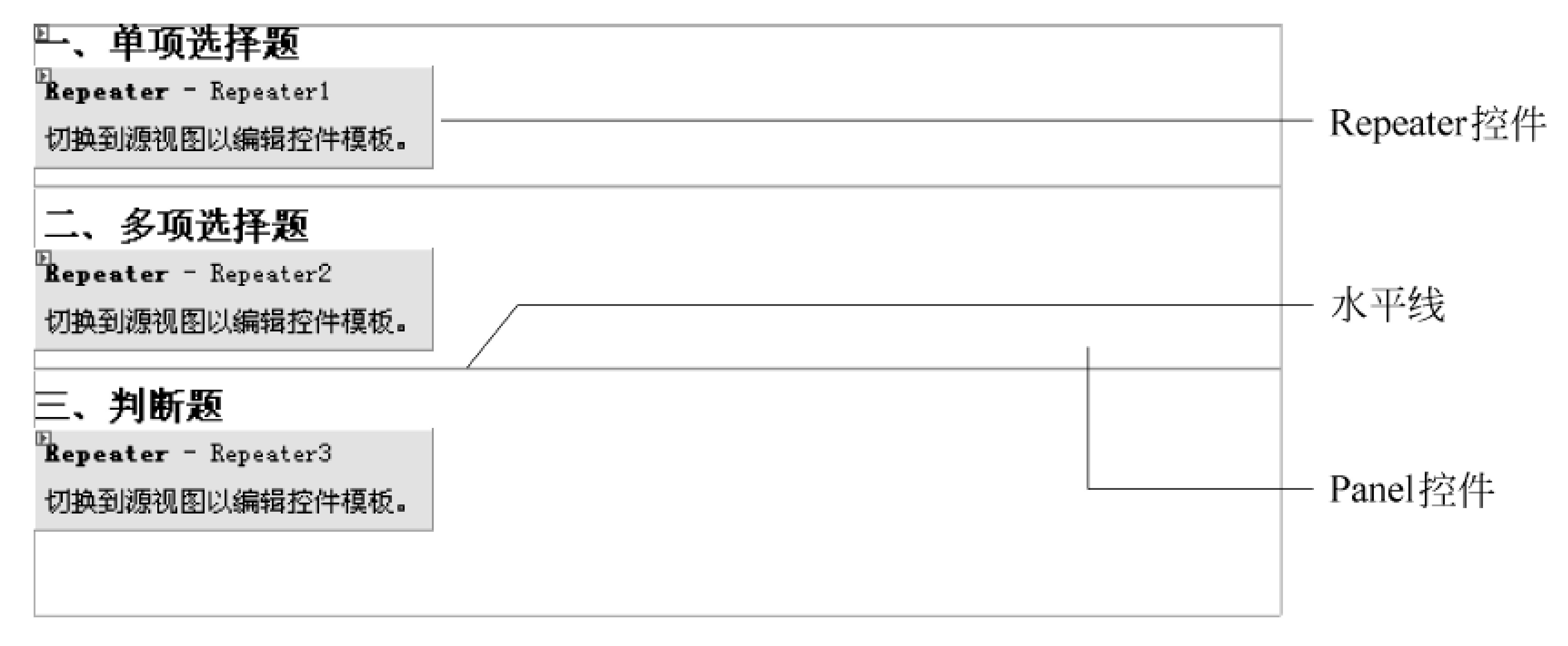


图 12-6 考试试题布局



为了便于理解和编程，我们指定 Panel 控件的 ID 值为 panelContent；第一个 Repeater 控件的 ID 值为 rpSingleQuestions，用来显示单项选择试题；第二个 Repeater 控件的 ID 值为 rpMultiQuestions，用来显示多项选择试题；第三个 Repeater 控件的 ID 值为 rpJudgeQuestions，用来显示判断试题。

2. 设置数据源

(1) 在数据访问层 AllQuestionsManager. cs 文件中添加方法随机选择试题。代码如下：

```
//随机选择试题
public static IList<examAllQuestion> GetExamQuestionRandom(int num,int sId,int typeId)
{
    string sql="select top"+num+" * from exam_allQuestions WHERE subject_id="+sId+"and
    question_type="+typeId+"order by newId()";
    return GetQuestionBySql(sql);
}
```

随机选择试题方法有三个参数，第一个参数 num 决定选择试题的数目，第二个参数 sId 决定选择什么科目的试题，第三个参数 typeId 决定选择的试题类型是单项选择题。随机抽出题库中的题目，方法很多，在此处我们使用 Order by newID()返回随机数据，也可以使用随机函数。

在随机选择试题的方法中调用了根据 SQL 语句查询试题信息的方法，代码如下：

```
//依据 SQL 语句查询试题
//不带参数
private static IList<examAllQuestion> GetQuestionBySql(string questionSql)
{
    List<examAllQuestion> questionList=new List<examAllQuestion>();
    //使用 using 指令及时释放资源
    using (DataTable table=ConnDBHelper.GetDataSet(questionSql))
    {
    foreach (DataRow row in table.Rows)
      {
          examAllQuestion examQuestion=new examAllQuestion();
          examQuestion.Question_id=(int)row["question_id"];
          examQuestion.Question_subject=(string)row["question_subject"];
          examQuestion.Question_keys=(string)row["question_keys"];
          examQuestion.A=Convert.ToString(row["a"]);
          examQuestion.B=Convert.ToString(row["b"]);
          examQuestion.C=Convert.ToString(row["c"]);
          examQuestion.D=Convert.ToString(row["d"]);
             examQuestion.examAllSubject = AllSubjectService.GetSubjectById((int) row
             ["subject_id"]);                        //外键
          examQuestion.examQuestionType=QuestionTypeService.GetQuestionTypeById
          ((int)row["question_type"]);               //外键
          questionList.Add(examQuestion);
      }
          return questionList;
```



```
    }
  }
```

(2) 用户界面表示层不能直接访问数据访问层的代码，需要在业务逻辑层添加如下代码。

```
//随机选择试题
public static IList< examAllQuestion>  GetExamQuestionRandom(int num,int sId, int typeId)
  {
      return AllQuestionsService.GetExamQuestionRandom(num,sId,typeId);
  }
```

(3) 在用户界面表示层的 DataBind 方法添加如下代码，为 rpSingleQuestions 控件配置数据源。

```
//为单项选择题指定数据源
rpSingleQuestions.DataSource=AllQuestionsManager.GetExamQuestionRandom(num1,sId,1);
rpSingleQuestions.DataBind();
```

多项选择题和判断题的数据源配置如下：

```
//多项选择题
rpMultiQuestions.DataSource=AllQuestions
Manager.GetExamQuestionRandom(num2,sId,2);
rpMultiQuestions.DataBind();
//判断题
rpJudgeQuestions.DataSource=AllQuestions
Manager.GetExamQuestionRandom(num3,sId,3);
rpJudgeQuestions.DataBind();
```

3. 试题显示

下面以单项选择题为例讨论试题的显示，其他两类试题类似。单项选择题的显示结果如图 12-7 所示，可以用无序列表标签
实现；每一行是一个列表项，用
标签实现；每一个小题是一个类别，用
实现。

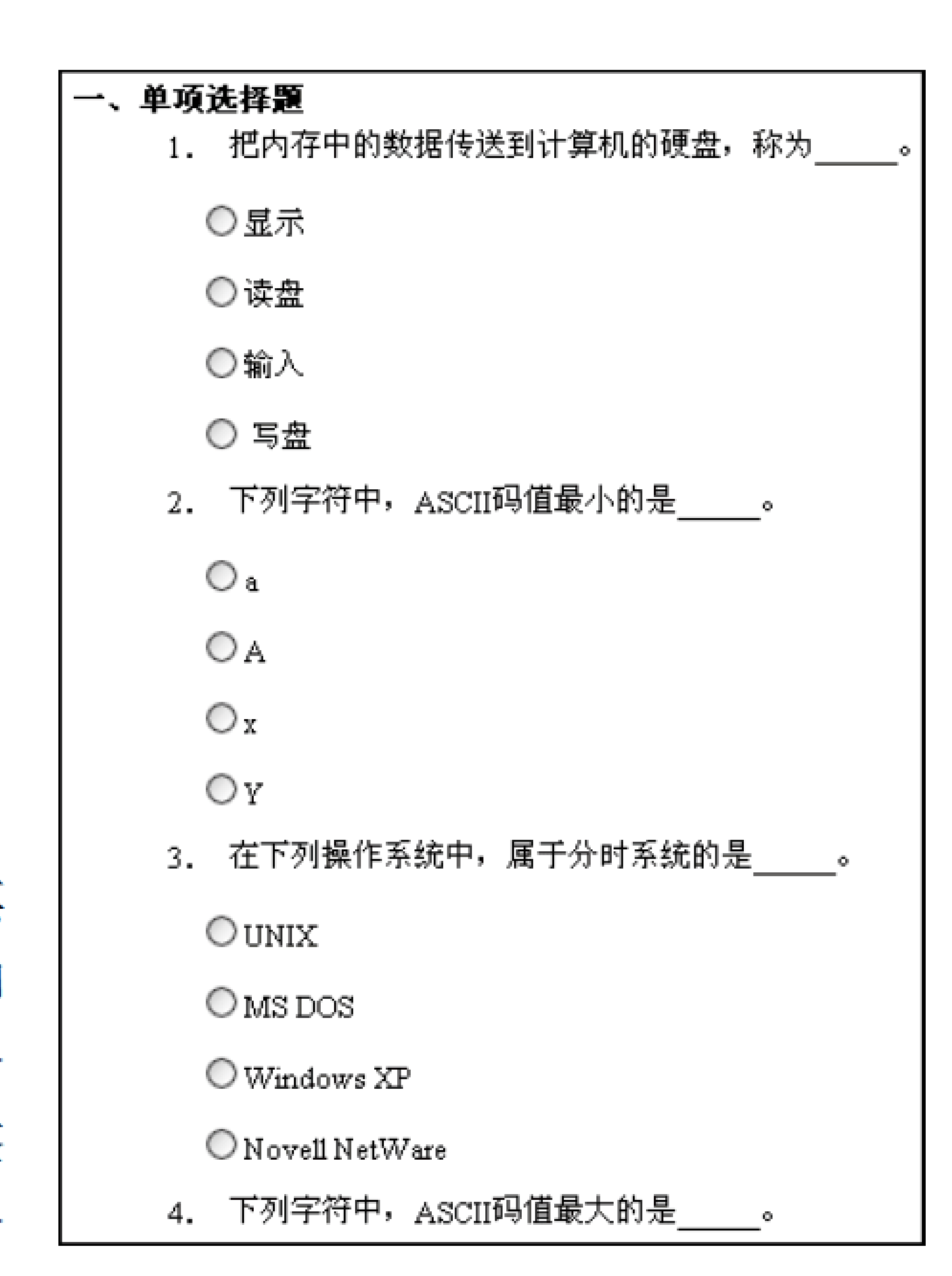


图 12-7　单项选择题

小贴士

由于每一小题是一个列表集，列表集需要重复显示(查询结果中每条记录对应一个列表集)，所以应当放在 Repeater 控件中的 ItemTemplate 模板内，不需要重复显示的内容(如每一种类型题的标题)，可以放在 HeaderTemplate 模板中。

(1) 在 Repeater 控件中添加<HeaderTemplate>模板，模板内添加每类题型的标题。代码如下：

```
<HeaderTemplate><strong>一、单项选择题</strong></HeaderTemplate>
```

(2) 在 Repeater 控件中添加<ItemTemplate>模板，在<ItemTemplate>模板内添加定义



每个小题的内容，需要显示的内容有题目前编号、题干和选择项。在<ItemTemplate>模板中添加一个
标签，代码如下：

```
<ItemTemplate>
<ul >
</ul>
</ItemTemplate>
```

(3) 在
模板中添加一个
标签，标签中添加 Label 标签用来显示题目的题干，代码如下：

```
<li><asp:Label ID="lblSubject1" runat="server" Text='<%# Eval("question_subject")%>'></
asp:Label></li>
```

<%#Eval("question_subject") %>绑定的是题干。

(4) 为了使每道小题前显示题号，在
标签中添加“<%#(Container.ItemIndex + 1)%>. ”，Container.ItemIndex 从 0 开始，每次加 1，代码如下：

```
<li><%# (Container.ItemIndex+1)%>. 
                        <asp:Label ID="lblSubject1" runat="server" Text='<%# Eval
                        ("question_subject")%>'></asp:Label>
</li>
```

(5) 在第一个
标签中再添加一个 Label 标签，用来存放当前显示试题在试题库中的 ID 值，试题的 ID 值不需要显示，但通过该 ID 值可以查找其他试题信息(例如试题答案)，为了不显示 ID 值，需要为 Label 标签添加 Visible 属性，属性值为 false。代码如下：

```
<asp:Label id="lblQid1" runat="server" Text='<%# Eval("question_id")%>'
    Visible="false">
</asp:Label>
```

(6) 在<ItemTemplate>模板中再添加 4 个
标签，用来显示试题选择项，在 4 个
标签中分别添加单选按钮标签 RadioButton，单选按钮的 Text 属性值为“单选项”，注意 4 个 RadioButton 标签的“GroupName 如”的值应一致。代码如下：

```
<asp:RadioButton id="rdoA" runat="server" Text='<%# Eval("a") %>' GroupName="S1">
</asp:RadioButton>
<asp:RadioButton id="rdoB" runat="server" Text='<%# Eval("b") %>'GroupName="S1">
</asp:RadioButton>
<asp:RadioButton id="rdoC" runat="server" Text='<%# Eval("c") %>' GroupName="S1">
</asp:RadioButton>
<asp:RadioButton id="rdoD" runat="server" Text='<%# Eval("d") %>' GroupName="S1">
</asp:RadioButton>
```

(7) 为了在列表中不显示项目符号，需要在 CSS 文件中添加以下代码。

```
li{
    list-style-type:none;
}
```



单项选择题显示完整代码段如下：

```
<asp:Repeater ID="rpSingleQuestions" runat="server">
  <HeaderTemplate><strong>一、单项选择题</strong></HeaderTemplate>
   <ItemTemplate>
     <ul>
<li><%#(Container.ItemIndex+1)%> 、
   <asp:Label ID="lblSubject1" runat="server" Text='<%#Eval("question_subject")%>'>
   </asp:Label>
   <asp:Label id="lblQid1" runat="server" Text='<%#Eval("question_id")%>' Visible="
   false">
        </asp:Label>
     </li>
       <li>    
            <asp:RadioButton id="rdoA" runat="server" Text='<%#Eval("a")%>'
            GroupName="S1">
          </asp:RadioButton>
        </li>
        <li>    
          <asp:RadioButton id="rdoB" runat="server" Text='<%#Eval("b")%>' GroupName="
          S1">
          </asp:RadioButton>
        </li>
        <li>    
          <asp:RadioButton id="rdoC" runat="server" Text='<%#Eval("c")%>' GroupName="
          S1">
          </asp:RadioButton>
        </li>
        <li>    
          <asp:RadioButton id="rdoD" runat="server" Text='<%#Eval("d")%>' GroupName="
          S1">
          </asp:RadioButton>
        </li>
     </ul>
     </ItemTemplate>
</asp:Repeater>
```

多项选择题与单项选择题类似，不同之处是多项选择题选择项用的是复选框 CheckBox。代码如下：

```
<asp:CheckBox id="CheckBox1" runat="server" Text='<%#Eval("a") %>'></asp:CheckBox>
```

多项选择题显示完整代码段如下：

```
<asp:Repeater ID="rpMultiQuestions" runat="server">
  <HeaderTemplate><strong>二、多项选择题</strong></HeaderTemplate>
  <ItemTemplate>
    <ul >
  <li ><%# (Container.ItemIndex+1)%>.  
       <asp:Label ID="lblSubject2" runat="server" Text='<%#Eval("question_subject") %>'>
       </asp:Label>
       <asp:Label id="lblQid2" runat="server" Text='<%#Eval("question_id") %>' Visible="
```



```
          False"></asp:Label>
          </li>
          <li>    <asp:CheckBox id="CheckBox1" runat="server" Text='<%
         #Eval("a") %>'></asp:CheckBox></li>
          <li>    <asp:CheckBox id="CheckBox2" runat="server" Text='<%
         #Eval("b") %>'></asp:CheckBox></li>
          <li>    <asp:CheckBox id="CheckBox3" runat="server" Text='<%
         #Eval("c") %>'></asp:CheckBox></li>
          <li>    <asp:CheckBox id="CheckBox4" runat="server" Text='<%
         #Eval("d") %>'></asp:CheckBox></li>
        </ul>
    </ItemTemplate>
</asp:Repeater>
```

判断题有对和错两个选项，实现方式同单项选择题。

判断题显示完整代码段如下：

```
<asp:Repeater ID="rpMultiQuestions"runat="server">
  <HeaderTemplate><strong>三、判断题</strong></HeaderTemplate>
  <ItemTemplate>
    <ul>
      <li><%# (Container.ItemIndex+1)%> 、
        <asp:Label ID="lblSubject3"runat="server"Text='<%#Eval("question_subject")%>'>
        </asp:Label>
        <asp:Label id="lblQid3"runat="server"Text='<%#Eval("question_id")%>'Visible="
        False">
        </asp:Label>
      </li>
      <li>    
          <asp:RadioButton id="rdoButton1"runat="server"Text='对' GroupName="S2">
            </asp:RadioButton>
               
            <asp:RadioButton id="rdoButton2"runat="server"Text='错'GroupName="S2">
            </asp:RadioButton>
        </li>
       </ul>
      </ItemTemplate>
</asp:Repeater>
```

12.2.3 保存考生试题信息

保存考生试题是为了以后查看考生的试卷和做题情况，当考生做完试题后单击“提交”按钮时，调用 btnSubmit_Click 事件，在该事件中调用三个方法 saveSingle(papertdName)、saveMulti(papertdName)和 saveJudge(papertdName)，这三个方法分别用来保存单选题、多选题和判断题。

下面着重介绍单选题的保存。保存单选题用一个 foreach 循环实现，每一次循环保存一道小题。首先从单选题的 Repeater 控件中取出一项 ItemTemplate 生成一个 RepeaterItem 实例 rp。





前面在显示试题时曾在<li>标签内放置了一个 Label 标签保存试题的 ID，通过试题 ID 利用方法 GetExamQuestionById()可以获取除了考生答案之外的其他试题信息。

获取考生答案用一个 if 语句实现，判断 rdoA 是否被选中，如果选中则考生答案就是"A"；否则再判断 rdoB 是否被选中，如果选中则考生答案就是"B"；否则再判断 rdoC 是否被选中，如果选中则考生答案就是"C"；否则再判断 rdoD 是否被选中，如果选中则考生答案就是"D"。

最后利用 AddQuestion()方法把考生的试题信息保存在试题表中。

代码如下：

```
//保存试卷单选题信息
private void saveSingle(string papertdName)
{
    foreach (RepeaterItem rp in rpSingleQuestions.Items)
    {
        int question_id=int.Parse(((Label)rp.FindControl("lblQid1")).Text);
        examAllQuestion userQuestion=AllQuestionsManager.GetExamQuestionById
        (question_id);
        //获取考生答案
        string user_key="";
        if (((RadioButton)rp.FindControl("rdoA")).Checked)
        {
            user_key="A";
        }
        else if (((RadioButton)rp.FindControl("rdoB")).Checked)
        {
            user_key="B";
        }
        else if (((RadioButton)rp.FindControl("rdoC")).Checked)
        {
            user_key="C";
        }
        else if (((RadioButton)rp.FindControl("rdoD")).Checked)
        {
            user_key="D";
        }
        examPaper userPaper=new examPaper();
        userPaper.Question_id=question_id;
        userPaper.Question_subject=userQuestion.Question_subject;
        userPaper.examAllSubject=userQuestion.examAllSubject;
        userPaper.examQuestionType=userQuestion.examQuestionType;
        userPaper.Question_keys=userQuestion.Question_keys;
        userPaper.A=userQuestion.A;
        userPaper.B=userQuestion.B;
        userPaper.C=userQuestion.C;
        userPaper.D=userQuestion.D;
        userPaper.User_key=user_key;
        ExamPaperManager.AddQuestion(userPaper, papertdName);
        if (user_key==userQuestion.Question_keys)
        {
            totalscore=totalscore+score1;
        }
```



```
        }
    }
```

AddQuestion()方法根据参数试题 ExamQuestion 对象和表名 paperName 把对象的相关属性添加到表中，代码如下：

```
//添加新试题
public static examPaper AddQuestion(examPaper examQuestion,string paperName)
{
    string sql = "insert "+paperName+"(subject_id,question_type,question_subject,question_
    keys,a,b,c,d,user_key)"+
    "VALUES (@subjectId,@questionType,@questionSubject,@questionKeys,@A,@B,@C,@D,@User
    _key)";
    sql+= " ; SELECT @@IDENTITY";
    SqlParameter[] para =new SqlParameter[]
      {

          new SqlParameter("@subjectId",examQuestion.examAllSubject.Subjectt_id),
                                                                          //外键
          new SqlParameter("@questionType",examQuestion.examQuestionType.Type_id),
                                                                          //外键
          new SqlParameter("@questionSubject",examQuestion.Question_subject),
          new SqlParameter("@questionKeys",examQuestion.Question_keys),
          new SqlParameter("@A",examQuestion.A),
          new SqlParameter("@B",examQuestion.B),
          new SqlParameter("@C",examQuestion.C),
          new SqlParameter("@D",examQuestion.D),
          new SqlParameter("@User_key",examQuestion.User_key)
      };
int newQuestionId =ConnDBHelper.GetScalar(sql,para);
return GetExamQuestionById(newQuestionId);
}
```

多选题、判断题的保存和单选题类似，代码如下：

```
//保存多选题信息
private void saveMulti(string papertdName)
{
    foreach(RepeaterItem rp in rpMultiQuestions.Items)
    {
        int question_id= int.Parse(((Label)rp.FindControl("lblQid2")).Text);
        examAllQuestion userQuestion= AllQuestionsManager.GetExamQuestionById(question_id);
        //获取考生答案
        string user_key= "";
        if(((CheckBox)rp.FindControl("CheckBox1")).Checked)
        {
            user_key+= "A";
        }
        if(((CheckBox)rp.FindControl("CheckBox2")).Checked)
        {
            user_key+=  "B";
        }
```




```
            if(((CheckBox)rp.FindControl("CheckBox3")).Checked)
            {
                user_key+="C";
            }
            if(((CheckBox)rp.FindControl("CheckBox4")).Checked)
            {
                user_key+="D";
            }
            examPaper userPaper=new examPaper();
            userPaper.Question_id=question_id;
            userPaper.Question_subject=userQuestion.Question_subject;
            userPaper.examAllSubject=userQuestion.examAllSubject;
            userPaper.examQuestionType=userQuestion.examQuestionType;
            userPaper.Question_keys=userQuestion.Question_keys;
            userPaper.A=userQuestion.A;
            userPaper.B=userQuestion.B;
            userPaper.C=userQuestion.C;
            userPaper.D=userQuestion.D;
            userPaper.User_key=user_key;
            ExamPaperManager.AddQuestion(userPaper,papertdName);
            if(user_key==userQuestion.Question_keys)

            {
                totalscore=totalscore+score2;
            }
        }
    }
    //保存试卷判断题信息
    private void saveJudge(string papertdName)
    {
        foreach(RepeaterItem rp in rpJudgeQuestions.Items)
        {
            int question_id=int.Parse(((Label)rp.FindControl("lblQid3")).Text);
            examAllQuestion userQuestion=AllQuestionsManager.GetExamQuestionById(question_id);
            //获取考生答案
            string user_key="";
            if(((RadioButton)rp.FindControl("rdoButton1")).Checked)
            {
                user_key="正确";
            }
            else if(((RadioButton)rp.FindControl("rdoButton2")).Checked)
            {
                user_key="错误";
            }
            examPaper userPaper=new examPaper();
            userPaper.Question_id=question_id;
            userPaper.Question_subject=userQuestion.Question_subject;
            userPaper.examAllSubject=userQuestion.examAllSubject;
            userPaper.examQuestionType=userQuestion.examQuestionType;
            userPaper.Question_keys=userQuestion.Question_keys;
            userPaper.User_key=user_key;
            ExamPaperManager.AddQuestion(userPaper,papertdName);
            if (user_key==userQuestion.Question_keys)
```





```
            {
                totalscore=totalscore+score3;
            }
        }
    }
```

12.2.4 考生成绩统计

1. 获取考生答案和总成绩

考生成绩统计的具体实现包含在保存试题的方法 saveSingle()、saveMulti()和 saveJudge()中，当得到试题的信息和考生答案后，判断试题信息中的正确答案和考生答案是否相同，如果相同，则总分加上该类题每道小题的分数。代码如下：

单选题：

```
if (user_key==userQuestion.Question_keys)
{
    totalscore=totalscore+score1;
}
```

多选题：

```
if (user_key==userQuestion.Question_keys)
{
    totalscore=totalscore+score2;
}
```

判断题：

```
if (user_key==userQuestion.Question_keys)
{
    totalscore=totalscore+score3;
}
```

totalscore 是一个全局变量，初始值为 0，score1、score2 和 score3 的值由 PapermgManager 对象的 GetPaperMessageBySidesId 方法得到，在单击"提交"按钮时获取，代码如下：

```
protected void btnSubmit_Click(object sender,ImageClickEventArgs e)
{
    string subjectId=Session["Lession"].ToString();
    int sId=Convert.ToInt32(subjectId);
    examPapermg paperMessage=PapermgManager.GetPaperMessageBySid(sId);
    score1=paperMessage.Score1;
    score2=paperMessage.Score2;
    score3=paperMessage.Score3;
}
```

2. 将总成绩保存在成绩统计表中

考生成绩总成绩保存由 saveScore 方法实现，单击"提交"按钮时调用。

saveScore 方法如下：



```
private void saveScore(string papertdName)
{
    examStuScore stuScore=new examStuScore();
    stuScore.Stu_id=stuId;

    stuScore.Paperdb_name=papertdName;
    DateTime dt=DateTime.Now;
    stuScore.Try_date=dt.ToShortDateString().ToString();
    stuScore.Stu_score=totalscore;
    StuScoreManager.AddExamStuScore(stuScore);
}
```

练　　习

1. 单项选择题

（1）以下数据控件中，________没有内置格式。

A. DataList　　B. GridView　　C. FormView　　D. Repeater

（2）要实现 Repeater 控件的编辑支持，应该在该控件的________事件中处理程序。

A. ItemCommand　　B. ItemCreated　　C. ItemDataBound　　D. EditCommand

（3）下列哪一个不是 Repeater 的模板？________

A. ItemDataBound　　B. ItemTemplate

C. HeaderTemplate　　D. AlternatingItemTemplate

（4）在 Repeater 控件中，________模板内可以有控件绑定数据源中的数据。

A. SeparatorTemplate　　B. FooterTemplate

C. HeaderTemplate　　D. AlternatingItemTemplate

2. 简答题

（1）Repeater 控件有哪些模板？分别说明它们的功能。

（2）在线考试系统中试题是怎样随机显示的？

实　　训

实训目的

（1）熟悉 Repeater 控件的使用。

（2）完善在线考试系统的试卷管理功能。

实训内容

（1）使用 Repeater 在表格中显示在线考试数据库中的学生信息，如图 12-8 所示。

（2）考试前，任课老师在考试前已经设置了科目考试信息（试卷中有哪些题型，每种题型多少道题，每道题多少分），并将这些信息保存在 exam_papermg 数据表中。完成这些功能。



学号	姓名	班级	密码
0000001	白嘉辉	1	0
0000002	武超帅	1	0
0000003	杨光宇	1	0
0000004	张西良	2	3
0000005	朱彬彬	1	0
0000006	朱志伟	1	0
0000007	赵磊	1	0
0000008	戴亮	1	0
0000009	张鹏飞	2	0

图 12-8　用 Repeater 控件显示学生信息

实训指导

实训指导 1：

（1）启动 Visual Studio 2005，新建一网站。

（2）在设计视图中，从工具箱中拖曳 Repeater 控件到编辑窗口。

（3）为 Repeater 控件新建数据源，配置的方式如图 12-2 所示。生成代码如下：

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString=
"<%$ ConnectionStrings:onLineExam1ConnectionString %> "
    SelectCommand="SELECT [stu_id],[stu_name],[class_id],[stu_pwd] FROM [exam_students]">
</asp:SqlDataSource>
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">
```

（4）在源视图中为 Repeater 控件添加 HeaderTemplate 模板、ItemTemplate 模板、AlternatingItemTemplate 模板 和 FooterTemplate 模板，代码如下：

```
<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">
        <HeaderTemplate>
        <table cellpadding='0' cellspacing='0' bordercolor="#3399FF" border="1" width="
        600px">
            <tr style="height:20px;">
                <td width="25%" align="center" >学号</td>
                <td width="25%" align="center" >姓名</td>
                <td width="25%" align="center" >班级</td>
                <td width="25%" align="center">密码</td>
            </tr>
        </HeaderTemplate>
        <ItemTemplate>
        <tr align=center style="height:20px;">
              <td width="25%" align="center" ><%# Eval("Stu_id") %></td>
              <td width="25%" align="center" ><%# Eval("Stu_name") %></td>
              <td width="25%" align="center"><%# Eval("Class_id") %></td>
              <td width="25%" align="center" ><%# Eval("Stu_pwd") %></td>
              </tr>
          </ItemTemplate>
          <FooterTemplate>
      </table>
          </FooterTemplate>
</asp:Repeater>
```





HeaderTemplate 模板中是项目符号开始不需要重复显示的内容，代码如下：

```
<HeaderTemplate>
    <table cellpadding='0' cellspacing='0' bordercolor="#3399FF" border="1" width="600px"
    >
      <tr style="height:20px;">
        <td width= "25%" align="center" >学号</td>
        <td width= "25%" align="center" >姓名</td>
        <td width= "25%" align="center" >班级</td>
        <td width= "25%" align="center">密码</td>
      </tr>
</HeaderTemplate>
```

ItemTemplate 模板中是需要重复显示的内容。

FooterTemplate 模板中是表格的结束标记。

实训指导 2：

如图 12-9 所示，从在线考试系统后台管理面板中选择科目试卷设置，弹出图 12-10 所示的科目试卷设置窗口。在科目试卷设置窗口中选择要设置的科目，填入考试时间，并设置单选题、多选题和判断题的题量和每个小题的分数。

如果某门课程已经设置过，则再次选择该科目时直接显示设置过的信息。具体实现过程参考在线考试系统中 SubjectExam. aspx 和 SubjectExam. aspx. cs 的代码。

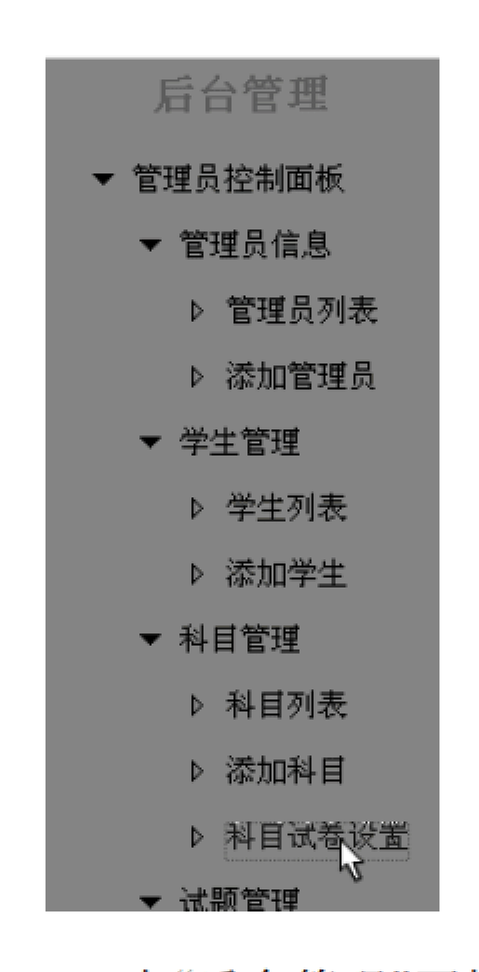


图 12-9　在“后台管理”面板中添加科目试卷设置



图 12-10　科目试卷设置页面

任务13　用户控件与网站版权

技能目标

能使用用户控件简化复杂开发。

知识目标

理解用户控件的特点,掌握用户控件的使用。

任务描述

使用用户控件开发在线考试系统网站版权。

本章任务如下:

- 了解及使用用户控件;
- 创建网站版权信息。

13.1　知识准备

下面介绍用户控件。

用户控件是转换成控件的 ASP.NET 页面,使用用户控件可实现页面中代码的重用。那么究竟什么是用户控件,它如何实现如此强大的功能呢?下面就来对此进行说明。

1. 了解用户控件

用户控件其实就是一种自定义的组合控件,通常由系统提供的可视化控件组合而成。用户控件中不仅可以定义显示界面,还可以编写事件处理代码。当多个网页中包括有部分相同的用户界面时,可以将这些内容相同的部分提取出来,做成用户控件。可以说,对用户控件可以像页面那样简单地去编辑,又可以像控件那样方便地去使用。

2. 创建用户控件

建立用户控件的第一步是建立一个.ascx 文件,这是用户控件需要的文件扩展名,如图 13-1 所示,它和创建页面一样简单。

可以看到这个用户控件的后台代码只有一行。

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="WebUserControl.ascx.cs"
Inherits="example_WebUserControl" %>
```

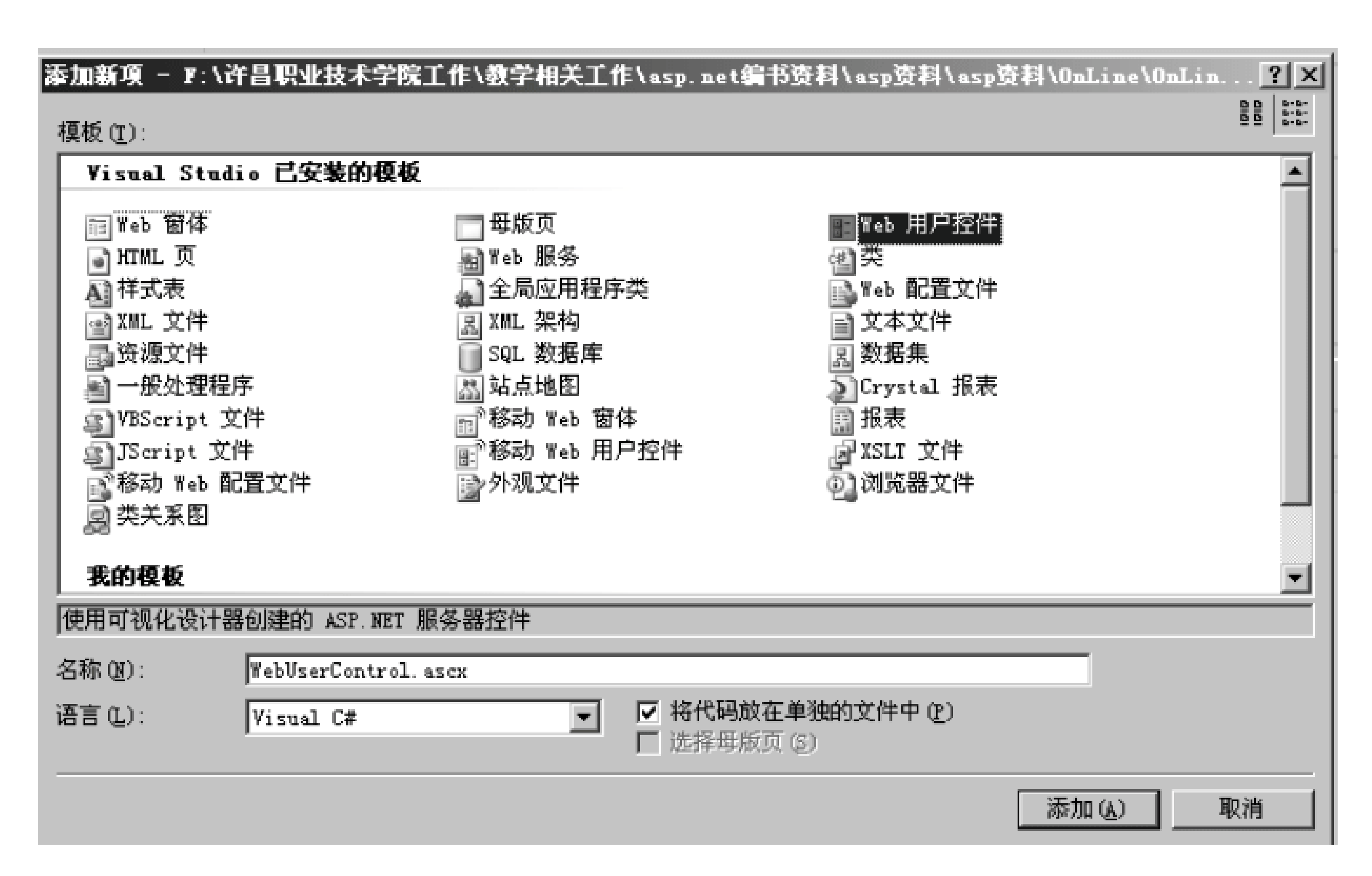


图 13-1　创建用户控件

此行指令与页面的@Page 指令非常相似，唯一不同的是，这里是@ Control 指令。

需要注意的是，在一个.ascx 文件中不能包含<head>、<form>及<body>标签，因为包含此.ascx 的页面已经包含了这些标签。

另外，与 Web 页面一样，用户控件也生成了它自己的.cs 文件。

```
public partial class example_WebUserControl:System.Web.UI.UserControl
{
    protected void Page_Load(object sender,EventArgs e)
    {
    }
}
```

在建立一个.ascx 文件后，我们可以像在页面上一样再向用户控件中拖入一些控件，如在 WebUserControl.ascx 中放置控件，如图 13-2 所示。

3. 使用用户控件

在 ASP.NET 中，使用用户控件也是一件很简单的事情，直接拖曳用户控件放入页面中的特定位置即可。如在项目的 shixunExamples 文件夹中新建一个页面 UseUserControl.aspx，在该页面的设计视图中直接把前面所建的用户控件拖入该页面，如图 13-3 所示。

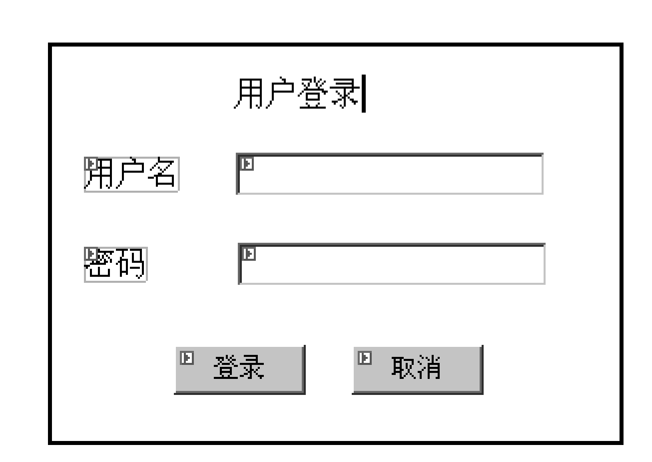


图 13-2　在 WebUserControl.ascx 中放置一些控件

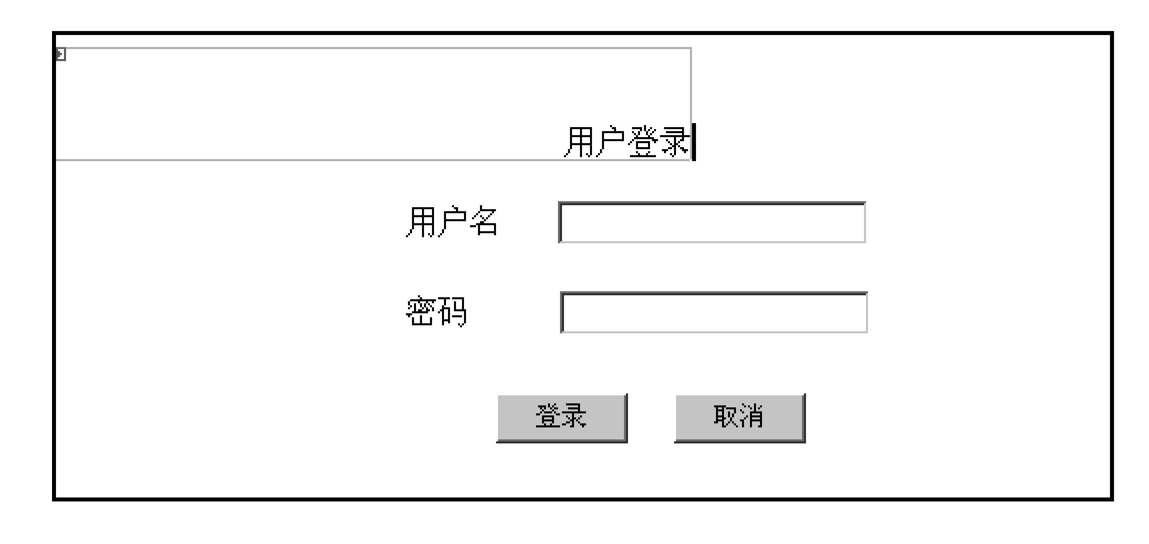


图 13-3　将用户控件拖入 Web 页面

下面我们来看一下对应代码。



```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="UseUserControl.aspx.cs" Inherits
="shixunExamples_UseUserControl" %>
<%@ Register Src="WebUserControl1.ascx" TagName="WebUserControl1" TagPrefix="uc2" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <uc2:WebUserControl1 ID="WebUserControl1_1" runat="server" />
         </div>
    </form>
</body>
</html>
```

可以看到，页面增加了一个@ Register 指令，该指令包括以下属性。

✍ TagPrefix：该属性定义了用户控件所使用的前缀，就是说它将前缀与用户控件相关联。此前缀将包括在用户控件元素的开始标记中。

✍ TagName：该属性定义了用户控件的名称，此名称将包括在用户控件元素的开始标记中。

✍ Src：该属性定义要包括在 Web 窗体页中的用户控件文件的虚拟路径。

4. 注意事项

(1) 用户控件可以包含其他用户控件。

用户控件可以包含其他的用户控件，但可能产生用户控件 1 包含用户控件 2，而用户控件 2 又包含了用户控件 1 的情况，就是说用户控件包含了自己，这是不允许的。在这种情况下 IDE 会自动检测到循环，提示错误，如图 13-4 所示。

呈现控件时出错 - WebUserControl2_1
检测到循环，UserControl 不能包含自己。

图 13-4　提示错误

(2) 用户控件不可以单独访问。

由于扩展名.ascx 是被禁止直接访问的类型，如果在地址栏中直接访问用户控件的地址，则会出现提示服务器错误的信息。

📖小贴士

有时用户控件要展示的功能已经在页面中的某一部分实现了，这时就可以将相关的代码复制过来。比如我们的登录页面已经做好了，可以通过这种方式完成用户控件的制作。

13.2　任务实施

13.2.1　使用用户控件创建网站版权信息

在试题管理系统中的后台管理员页面，添加用户控件 Copyright.ascx。在该用户控件中放入一个 Label 控件，代码如下：

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Copyright.ascx.cs" Inherits="
```



```
admin_Copyright" %>
<div>
    <asp:Label ID="LblAsUserControl" runat="server" Text="copyright&copy;2009 清华大学出
    版社" ></asp:Label></div>
```

该页面的布局如图 13-5 所示。



图 13-5　Copyright.ascx 的内容

13.2.2　在模板页中使用用户控件

1. 前台模板页使用用户控件

在项目 OnLineWeb 前台的模板页中，把用户控件 Copyright.ascx 拖入到 MasterPage.master 页面中，如图 13-6 所示。



图 13-6　在项目前台模板页中使用用户控件

2. 后台模板页使用用户控件

同样，在项目 OnLineWeb 后台的模板页中，把用户控件 Copyright.ascx 拖入到 AdminMasterPage.master 页面中，如图 13-7 所示。

在前台和后台的模板页中放置该用户控件后，所有使用模板页的页面都会显示模板页中的版权信息。



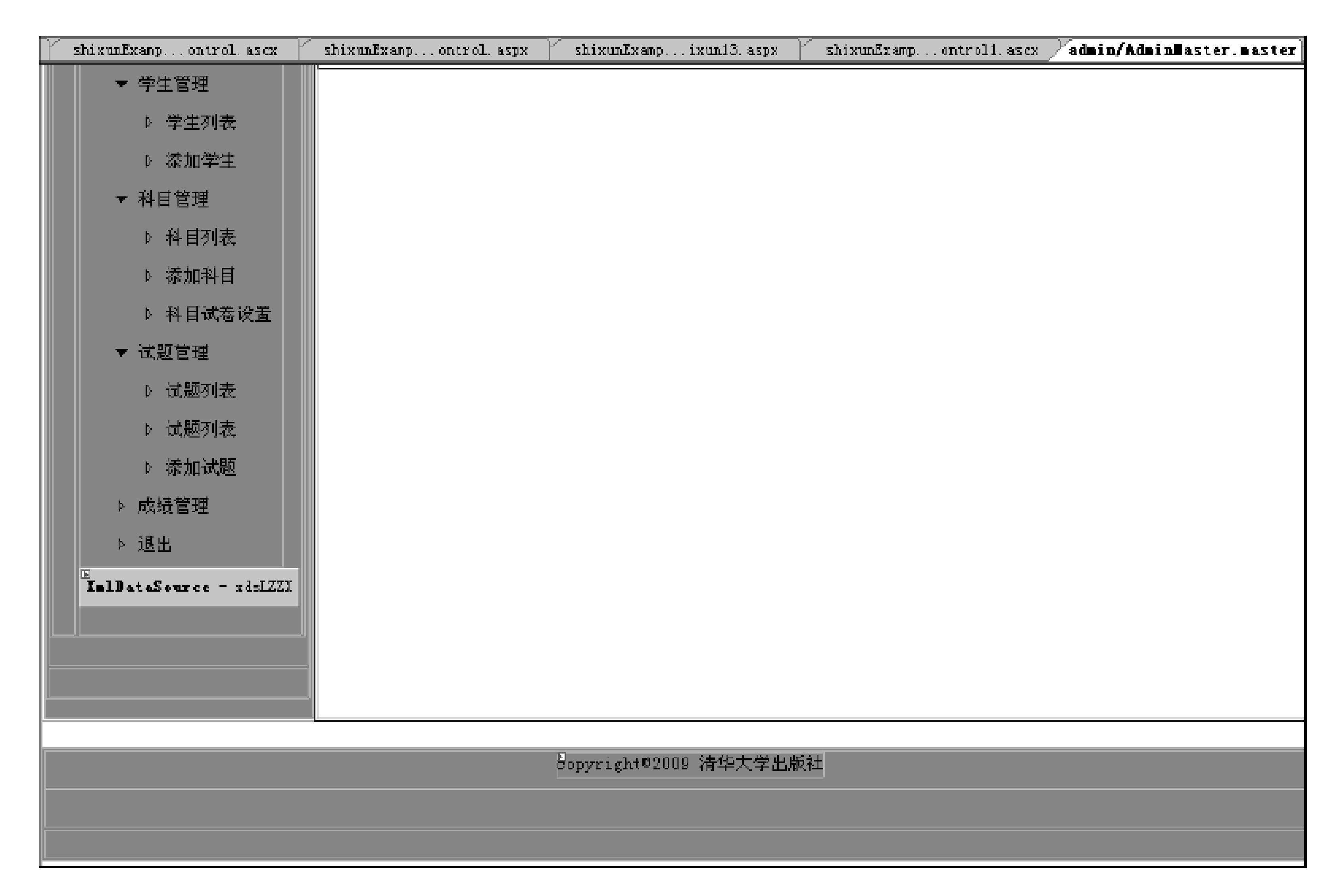


图 13-7　在项目后台模板页中使用用户控件

练　　习

单项选择题

(1) 有关用户控件的说法正确的是________。

A. 用户控件是一种自定义的组合控件

B. 用户控件不能在同一应用程序的不同网页上重用

C. 使用用户控件,需要像第三方控件一样在项目中注册该控件

D. 用户控件不能包含其他用户控件

(2) 保存用户控件需要使用的扩展名是________。

A. .aspx　　B. .asmx　　C. .ascx　　D. .asax

(3) ________指令用于使用已创建的自定义控件。

A. Register　　B. Control　　C. TagPrefix　　D. Import

实　　训

实训目的

(1) 熟练掌握创建用户控件的方法。

(2) 掌握使用用户控件的方法。

实训内容

(1) 创建一个用户登录的用户控件。

(2) 使用该用户控件完成后台管理员登录功能。



实训指导

（1）在实训项目中新建一个用户控件 WebUserControl1.ascx，该用户控件界面如图 13-8 所示。

（2）新建一个窗体页面 shixun13.aspx，在该页面中，将上面建立的用户控件 WebUserControl1.ascx 拖入该页面，即可直接运行 shixun13.aspx 页面的登录功能。

图 13-8　用户控件界面

（3）WebUserControl1.ascx 中的页面代码如下：

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="WebUserControl1
.ascx.cs" Inherits="example_WebUserControl" %>
<div>
    <div style="z-index: 106; left: 366px; background-image: url(image/yonghuguanli.
    jpg);width: 344px; position: absolute; top: 114px; height: 199px; background-color: #
    ccffcc">
<asp:Button ID="Button3" runat="server" Height="25px" Style="z-index: 100; left: 81px;
position: absolute; top: 136px" Text="登录" Width="100px" />
<asp:Button ID="Button4" runat="server" Height="25px" Style="z-index: 101; left: 194px;
position: absolute; top: 135px" Text="取消" Width="100px" />
<asp:Label ID="Label3" runat="server" Style="z-index: 102;left: 70px; position: absolute;
top: 45px" Text="用户名" Width="60px"></asp:Label>
<asp:TextBox ID="TextBox3" runat="server" Style="z-index: 103; left: 154px; position:
absolute;top: 44px"></asp:TextBox>
<asp:Label ID="Label4" runat="server" Style="z-index: 104; left: 70px; position: absolute;
top:91px" Text="密　码" Width="54px"></asp:Label>
<asp:TextBox ID="TextBox4" runat="server" Style="z-index: 106; left: 154px; position:
absolute; top: 87px"></asp:TextBox>
                       
                   用户登录</div>
</div>
```

shixun13.aspx 中的页面代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="shixun13.aspx.cs"
Inherits="shixunExamples_shixun13" %>
<%@ Register Src="WebUserControl1.ascx" TagName="WebUserControl1" TagPrefix="uc1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
      <uc1:WebUserControl1 ID="WebUserControl1_1" runat="server" />
      </div>
    </form>
</body>
</html>
```



任务14　文件及文件夹操作

技能目标

能使用 FileUpload 控件实现文件的上传和下载，能使用各种对象实现文件和目录操作。

知识目标

掌握 FileUpload 控件的用法，掌握文件读写的方法，掌握文件上传和下载的方法，掌握文件和目录操作的方法，掌握 XML 文件操作的方法。

任务描述

在线考试系统中并没有文件和目录操作，本章内容主要是为了完善 ASP.NET 知识体系。

本章任务如下：

- 了解文件的使用方法；
- 了解文件夹（目录）的操作方法。

14.1　文　　件

14.1.1　文件概述

文件是一个在逻辑上具有完整意义的一组相关信息的有序集合。计算机系统中的信息，如系统程序、标准子程序、应用程序和各种类型的数据，通常都以文件的形式保存在外存中。

文件可以是文本文档、图片、程序等。文件通常具有三个字母的文件扩展名，用于指示文件类型（例如，图片文件常常以 JPEG 格式保存并且扩展名为.jpg）。

14.1.2　文件的上传和下载

1. FileUpload 控件

在基于 Web 的应用程序中，经常需要允许用户把文件上传到 Web 服务器或从 Web 服务器上下载文件，ASP.NET 2.0 提供了 FileUpload 控件帮助完成这些操作。FileUpload 控件在工具箱中的图标为 FileUpload，该控件使用户可以更容易地浏览和选择用于上传的文件，它包含一个“浏览”按钮和用于输入文件名的文本框。只要用户在文本框中输入了完全限定的文件名，无论是直接输入或通过“浏览”按钮选择，都可以调用FileUpload的SaveAs

方法保存到磁盘上。

FileUpload 控件常用的属性如下。

✍ FileContent：返回一个指向上传文件的流对象(stream 类型)。

✍ FileName：返回要上传文件的名称，不包含路径信息(string 类型)。

✍ HasFile：如果是 true，则表示该控件有文件要上传(Boolean 类型)。

✍ PostedFile：返回已经上传文件的引用(HttpPostedFile 类型)。

在这 4 个属性中，PostedFile 属性是 HttpPostedFile 类型的。HttpPostedFile 类还具有以下几种属性。

✍ ContentLength：返回上传文件的按字节表示的文件大小(integer 类型)。

✍ ContentType：返回上传文件的 MIME 内容类型(string 类型)。

✍ FileName：返回文件在客户端的完全限定名(string 类型)。

✍ InputStream：返回一个指向上传文件的流对象(stream 类型)。

2. 文件的上传

文件上传通常都是利用 FileUpload 控件来完成的，在上传过程中通常需要记录上传文件名、大小、路径等信息。下面看一个文件上传的例子。

(1) Upload.aspx 页面核心代码

```
<body>
<form id="form1" runat="server">
<div>
<asp:FileUpload ID="FileUpload1" runat="server" Width="325px" />
<asp:Button ID="Button1" runat="server" onClick="Button1_Click" Text="上传" Width="44px"
/><br />
上传文件名:<asp:Label ID="Label2" runat="server"></asp:Label><br />
上传文件大小:<asp:Label ID="Label3" runat="server"></asp:Label><br />
上传文件路径及名:<asp:Label ID="Label4" runat="server"></asp:Label><br />
<asp:Label ID="Label1" runat="server" ForeColor="Red"></asp:Label><br />
</div>
</form>
</body>
```

(2) Upload.aspx.cs 页面核心代码

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;
public partial class _Default:System.Web.UI.Page
{
protected void Page_Load(object sender,EventArgs e)
{
```



```
}
protected void btnUpload_Click(object sender,EventArgs e)
{
if (FileUpload1.HasFile)                          //判断 FileUpload 控件中是否存在文件
{
    Label2.Text=FileUpload1.FileName;             //保存上传文件的名字
    Label3.Text=FileUpload1.PostedFile.ContentLength.ToString()+"字节";
                                                  //保存上传文件的大小
    Label4.Text=FileUpload1.PostedFile.FileName;
                                                  //保存上传文件的路径和文件名
    string extension=Path.GetExtension(FileUpload1.FileName);
                                                  //获得扩展名
    if (extension==".rar" || extension==".doc" || extension==".xls")
                                                  //限定文件类型
    {
        if (FileUpload1.PostedFile.ContentLength<10000000)
                                                  //判定文件大小是否小于 10MB
    {
        FileUpload1.SaveAs(Server.MapPath("~/upload/"+FileUpload1.FileName));
                                              //将文件保存在当前网站的 upload 文件夹下
        Label1.Text="恭喜你,上传成功!";
    }
    else
    {
        Label1.Text="上传失败,不是管理员,上传文件不能大于 10MB!";
    }
    }
  else
  {
      Label1.Text="上传失败,只能上传扩展名是.rar、.doc、.xls 的文件!";
  }
  else
  {
      Label1.Text="上传失败,或没指定正确的文件!";
  }
}
}
}
```

（3）运行效果

运行效果如图 14-1 所示。

3．文件的下载

代码如下：

```
using System;
using System.Configuration;
using System.Data;
using System.Web;
using System.Web.Security;
```





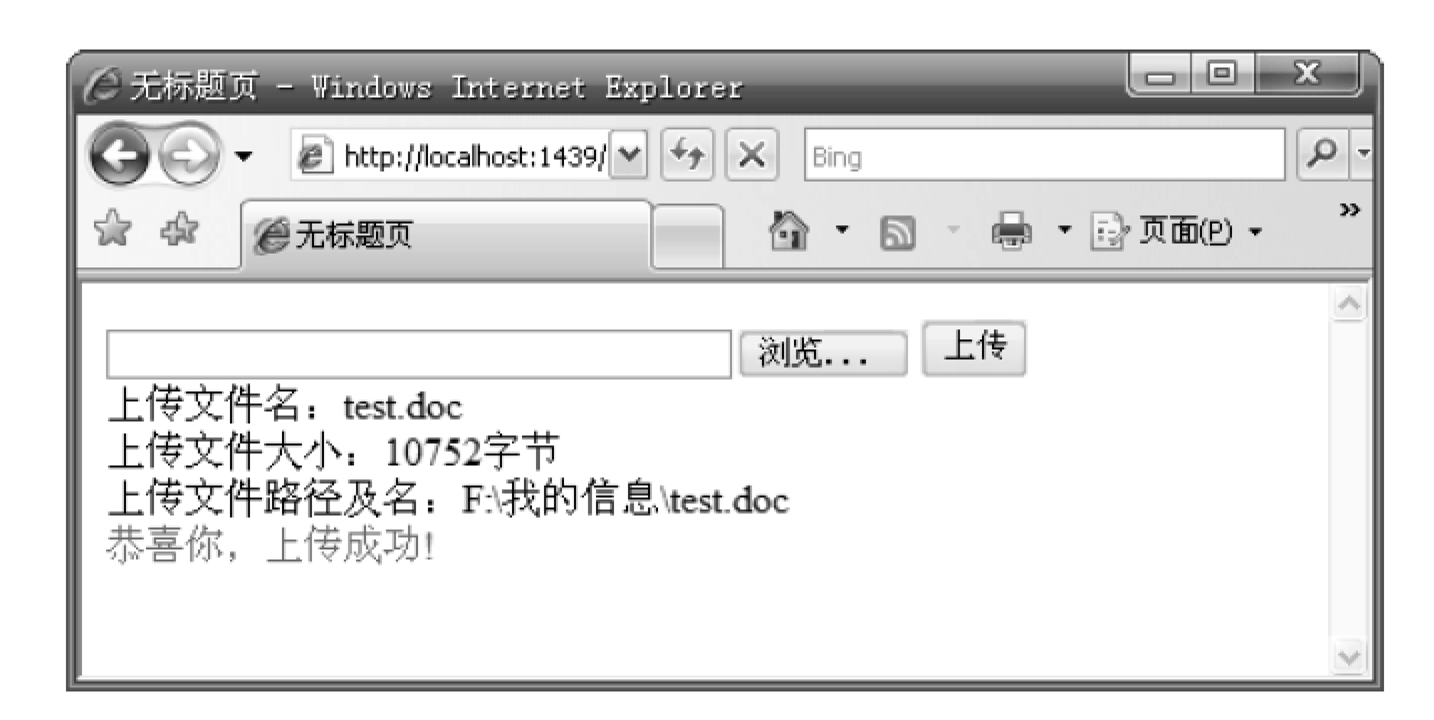


图 14-1　上传文件效果图

```
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using Microsoft.Win32;
using System.IO;
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender,EventArgs e)
    {
        DownloadFile("DownLoad\\aa.rar");
    }
protected void DownloadFile(string filename)
{
  string saveFileName="test.xls";
  int intStart=filename.LastIndexOf("\\") +1;
  saveFileName=filename.Substring(intStart,filename.Length-intStart);
  System.IO.FileInfo fi=new System.IO.FileInfo(filename);
  string fileextname=fi.Extension;
  string DEFAULT_CONTENT_TYPE="application/unknown";
  RegistryKey regkey,fileextkey;
  string filecontenttype;
  try
  {
      regkey=Registry.ClassesRoot;
      fileextkey=regkey.OpenSubKey(fileextname);filecontenttype=fileextkey.GetValue("Content
      Type",DEFAULT_CONTENT_TYPE).ToString();
  }
  catch
  {
      filecontenttype=DEFAULT_CONTENT_TYPE;
  }
  Response.Clear();
  Response.Charset="utf-8";
  Response.Buffer=true;
  this.EnableViewState=false;
  Response.ContentEncoding=System.Text.Encoding.UTF8;
  Response.AppendHeader("Content-Disposition","attachment;filename=" +saveFileName);
```



```
        Response.ContentType=filecontenttype;
        Response.WriteFile(filename);
        Response.Flush();
        Response.Close();
        Response.End();
    }
    //TransmitFile 实现下载 rar 文件
    protected void Button1_Click(object sender,EventArgs e)
    {
        /* 微软为 Response 对象提供了一个新的方法 TransmitFile 来解决使用 Response.BinaryWrite
           下载超过 400MB 的文件时导致 Aspnet_wp.exe 进程被回收而无法成功下载的问题。代码如
           下:*/
        Response.ContentType="application/x-zip-compressed";
        Response.AddHeader("Content-Disposition","attachment;filename=z.zip");
        string filename=Server.MapPath("Download/aa.rar");
        Response.TransmitFile(filename);
    }
    //WriteFile 实现下载
    protected void Button2_Click(object sender,EventArgs e)
    {
        /*
        using System.IO;

        */
            string fileName="asd.txt";                              //客户端保存的文件名
            string filePath=Server.MapPath("Download/huishouzhan.txt");        //路径
            FileInfo fileInfo=new FileInfo(filePath);
            Response.Clear();
            Response.ClearContent();
            Response.ClearHeaders();
            Response.AddHeader("Content-Disposition","attachment;filename="+fileName);
            Response.AddHeader("Content-Length",fileInfo.Length.ToString());
            Response.AddHeader("Content-Transfer-Encoding","binary");
            Response.ContentType="application/octet-stream";
            Response.ContentEncoding=System.Text.Encoding.GetEncoding("gb2312");
            Response.WriteFile(fileInfo.FullName);
            Response.Flush();
            Response.End();
    }
        //WriteFile 分块下载
        protected void Button3_Click(object sender,EventArgs e)
        {
            string fileName="aaa.txt";                              //客户端保存的文件名
            string filePath=Server.MapPath("Download/huishouzhan.txt");        //路径
            System.IO.FileInfo fileInfo=new System.IO.FileInfo(filePath);
            if (fileInfo.Exists==true)
            {
                const long ChunkSize=102400;
                                  //每次读取文件,只读取 100KB,这样可以缓解服务器的压力
                byte[] buffer=new byte[ChunkSize];
                Response.Clear();
```



```
                System.IO.FileStream iStream=System.IO.File.OpenRead(filePath);
                long dataLengthToRead=iStream.Length;            //获取下载的文件总大小
                Response.ContentType="application/octet-stream";
                Response.AddHeader("Content-Disposition","attachment; filename=
                " +HttpUtility.UrlEncode(fileName));
                while (dataLengthToRead >0 && Response.IsClientConnected)
                {
                    int lengthRead=iStream.Read(buffer,0,Convert.ToInt32(ChunkSize));
                                                                  //读取的大小
                    Response.OutputStream.Write(buffer,0,lengthRead);
                    Response.Flush();
                    dataLengthToRead=dataLengthToRead - lengthRead;
                }
                Response.Close();
            }
        }
        //<summary>
        //流方式下载
        //</summary>
        //<param name="sender"></param>
        //<param name="e"></param>
            protected void Button4_Click(object sender,EventArgs e)
            {
                string fileName="aaa.txt";                          //客户端保存的文件名
                string filePath=Server.MapPath("Download/huishouzhan.txt");      //路径
                //以字符流的形式下载文件
                FileStream fs=new FileStream(filePath,FileMode.Open);
                byte[] bytes=new byte[(int)fs.Length];
                fs.Read(bytes,0,bytes.Length);
                fs.Close();
                Response.ContentType="application/octet-stream";
                //通知浏览器下载文件而不是打开
                 Response.AddHeader("Content-Disposition","attachment; filename=
                 "+HttpUtility.UrlEncode(fileName,System.Text.Encoding.UTF8));
                Response.BinaryWrite(bytes);
                Response.Flush();
                Response.End();
            }
    }
```

14.2　文件和目录操作

在 ASP.NET 中,文件的基本操作包括:文件的创建、移动、复制、删除和读写等;目录的操作包括目录的创建等。

14.2.1　文件操作

有关文件的各种操作主要是通过 File 和 FileInfo 类来完成的,在使用时需要引用



System.IO命名空间。File类提供用于创建、复制、删除、移动和打开文件的静态方法,并协助创建FileStream对象;FileInfo类提供创建、复制、删除、移动和打开文件的实例方法,并且帮助创建FileStream对象。这两个类之间的主要区别在于:File类是一个静态类,它对应整个文件系统进行操作,方法均为静态方法;FileInfo类是一个实例类,它对应某一个文件进行操作,方法大部分为实例方法,它的操作有可能是调用的File中的对应静态方法。File类操作起来比较方便,有很多时候不像生成一个FileInfo对象那么麻烦。File类和FileInfo类包括的常用方法基本类似,但是使用FileInfo类的方法必须首先实例化一个FileInfo类对象。下面介绍File类的主要方法。

1. 文件打开方法:File.Open

(1) 声明

```
public static FileStream Open(string path,FileMode mode)
```

其中,path表示文件路径,mode表示文件打开方式。

(2) 示例

下面的代码打开存放在C:\Upload目录下名称为File1.txt的文件,并在该文件中写入hello world。

```
private void OpenFile()
{
    FileStream.TextFile=File.Open(@"C:\Upload\File1.txt",FileMode.Append);
    byte[] Info={(byte)'h',(byte)'e',(byte)'l',(byte)'l',(byte)'o',(byte)'w',(byte)'o',
    (byte)'r',(byte)'l',(byte)'d'};
    TextFile.Write(Info,0,Info.Length);
    TextFile.Close();
}
```

2. 文件创建方法:File.Create

(1) 声明

```
public static FileStream Create(string path);
```

(2) 示例

下面的代码演示如何在C:\Upload下创建名为File1.txt的文件。

由于File.Create方法默认向所有用户授予对新文件的完全读/写访问权限,所以文件是用读/写访问权限打开的,必须关闭后才能由其他应用程序打开。所以需要使用FileStream类的Close方法将所创建的文件关闭。代码如下:

```
private void MakeFile()
{
    FileStream NewText=File.Create(@ "C:\Upload\File1.txt");
    NewText.Close();
}
```



3. 文件删除方法：File. Delete

(1) 声明

```
public static void Delete(string path);
```

(2) 示例

下面的代码演示如何删除 C:\Upload 目录下的 File1. txt 文件。

```
private void DeleteFile()
{
    File.Delete(@"C:\Upload\File1.txt");
}
```

4. 文件复制方法：File. Copy

(1) 声明

```
public static void Copy(string sourceFileName,string destFileName,bool overwrite);
```

(2) 示例

下面的代码将 C:\Upload\File1. txt 复制到 C:\Upload\BackUp. txt。由于 Cope 方法的 OverWrite 参数设为 true，所以如果 BackUp. txt 文件已存在的话，将会被复制过去的文件所覆盖。

```
private void CopyFile()
{
    File.Copy(@"C:\Upload\File1.txt",@"C:\Upload\BackUp.txt",true);
}
```

5. 文件移动方法：File. Move

(1) 声明

```
public static void Move(string sourceFileName,string destFileName);
```

(2) 示例

下面的代码可以将 C:\Upload 下的 BackUp. txt 文件移动到 C 盘根目录下。注意：只能在同一个逻辑盘下进行文件转移。如果试图将 C 盘下的文件转移到 D 盘，将发生错误。

```
private void MoveFile()
{
    File.Move(@"C:\Upload\BackUp.txt",@"C:\BackUp.txt");
}
```

6. 设置文件属性方法：File. SetAttributes

(1) 声明

```
public static void SetAttributes(string path,FileAttributes fileAttributes);
```



(2) 示例

下面的代码可以设置文件 C:\Upload\File1.txt 的属性为只读、隐藏。

```
private void SetFile()
{
    File.SetAttributes(@"C:\Upload\File1.txt", FileAttributes.ReadOnly
    FileAttributes.Hidden);
}
```

文件除了常用的只读和隐藏属性外,还有 Archive(文件存档状态)、System(系统文件)、Temporary(临时文件)等。

7. 判断文件是否存在的方法:File.Exist

(1) 声明

```
public static bool Exists(string path);
```

(2) 示例

下面的代码判断是否存在 C:\Upload\File1.txt 文件。若存在,先复制该文件,然后将其删除,最后将复制的文件移动;若不存在,则先创建该文件,然后打开该文件并进行写入操作,最后将文件属性设为只读、隐藏。

```
if(File.Exists(@"C:\Upload\File1.txt"))        //判断文件是否存在
{
  CopyFile();                                  //复制文件
  DeleteFile();                                //删除文件
  MoveFile();                                  //移动文件
}
else
{
  MakeFile();                                  //生成文件
  OpenFile();                                  //打开文件
  SetFile();                                   //设置文件属性
}
```

8. 操作 Text 文本

对文本文件的操作可以通过以下方法来实现。

✍ AppendText:将文本追加到现有文件。

✍ CreateText:为写入文本创建或打开新文件。

✍ OpenText:打开现有文本文件以进行读取。

但上述方法主要是对 UTF-8 的编码文本进行操作,不够灵活。这里介绍下面的代码完成对 txt 文件的操作。

(1) 对 txt 文件进行"读"操作

示例代码如下:

```
StreamReader TxtReader= new StreamReader(@"C:\Upload\File1.txt",System.Text.Encoding.
Default);
string FileContent;
```



```
FileContent=TxtReader.ReadEnd();
TxtReader.Close();
```

（2）对 txt 文件进行“写”操作

示例代码如下：

```
StreamWriter=new StreamWrite(@"C:\Upload\File1.txt",System.Text.Encoding.Default);
string FileContent;
TxtWriter.Write(FileContent);
TxtWriter.Close();
```

14.2.2　目录操作

在 ASP.NET 中对目录的操作是通过 Directory 类和 DirectoryInfo 类来完成的，使用时需要添加 System.IO 命名空间。Directory 类公开用于创建、移动和枚举与目录与子目录相关的静态方法；DirectoryInfo 类公开用于创建、移动和枚举目录与子目录的实例方法。这两个类的区别与 File 类和 FileInfo 类的区别相同。

1. 目录创建方法：Directory.CreateDirectory

（1）声明

```
public static DirectoryInfo CreateDirectory(string path);
```

（2）示例

下面的代码演示在 C:\Upload 文件夹下创建名为 NewDirectory 的目录。

```
private void MakeDirectory()
{
    Directory.CreateDirectory(@"C:\Upload\NewDirectoty");
}
```

2. 目录属性设置方法：DirectoryInfo.Atttributes

下面的代码设置 C:\Upload\NewDirectory 目录为只读、隐藏。与文件属性相同，目录属性也是使用 FileAttributes 来进行设置的。

```
private void SetDirectory()
{
    DirectoryInfo NewDirInfo=new DirectoryInfo(@"C:\tempuploads\NewDirectoty");
    NewDirInfo.Atttributes=FileAttributes.ReadOnly FileAttributes.Hidden;
}
```

3. 目录删除方法：Directory.Delete

（1）声明

```
public static void Delete(string path,bool recursive);
```

（2）示例



下面的代码可以将 C:\Upload\BackUp 目录删除。Delete 方法的第二个参数为 bool 类型,它可以决定是否删除非空目录。如果该参数值为 true,将删除整个目录,即使该目录下有文件或子目录;若为 false,则仅当目录为空时才可删除。

```
private void DeleteDirectory()
{
    Directory.Delete(@"C:\Upload\BackUp",true);
}
```

4. 目录移动方法:Directory.Move

(1) 声明

```
public static void Move(string sourceDirName,string destDirName);
```

(2) 示例

下面的代码将目录 C:\Upload\NewDirectory 移动到 C:\Upload\BackUp。

```
private void MoveDirectory()
{
    File.Move(@"C:\Upload\NewDirectory",@"C:\Upload\BackUp");
}
```

5. 获取当前目录下的所有子目录方法:Directory.GetDirectories

(1) 声明

```
public static string[] GetDirectories(string path;);
```

(2) 示例

下面的代码读出 C:\Upload\目录下的所有子目录,并将其存储到字符串数组中。

```
private void GetDirectory()
{
    string[]Directorys;
    Directorys=Directory.GetDirectories (@"C:\Upload");
}
```

6. 获取当前目录下的所有文件方法:Directory.GetFiles

(1) 声明

```
public static string[] GetFiles(string path;);
```

(2) 示例

下面的代码读出 C:\Upload\目录下的所有文件,并将其存储到字符串数组中。

```
private void GetFile()
{
    string[]Files;
    Files=Directory. GetFiles (@"C:\Upload",);
```



```
}
```

7. 判断目录是否存在方法：Directory. Exist

（1）声明

```
public static bool Exists(string path;);
```

（2）示例

下面的代码判断是否存在 C:\Upload\NewDirectory 目录。若存在，先获取该目录下的子目录和文件，然后将其移动，最后将移动后的目录删除；若不存在，则先创建该目录，然后将目录属性设为只读、隐藏。

```
if(File.Exists(@"C:\Upload\NewDirectory"))         //判断目录是否存在
{
    GetDirectory();                                 //获取子目录
    GetFile();                                      //获取文件
    MoveDirectory();                                //移动目录
    DeleteDirectory();                              //删除目录
}
else
{
    MakeDirectory();                                //生成目录
    SetDirectory();                                 //设置目录属性
}
```

小贴士

路径有 3 种方式，分别为当前目录下的相对路径、当前工作盘的相对路径、绝对路径。以 C:\Tmp\Book 为例(假定当前工作目录为 C:\Tmp)。Book(当前目录下的相对路径)、\Tmp\Book(当前工作盘的相对路径)、C:\Tmp\Book(绝对路径)都表示 C:\Tmp\Book。

另外，在 C# 中 “\”是特殊字符，要表示它需要使用“\\”。由于这种写法不方便，C# 语言提供了@对其简化。只要在字符串前加上@即可直接使用“\”。所以上面的路径在 C# 中应该表示为"Book"、@"\Tmp\Book"、@"C:\Tmp\Book"。

14.3　XML 文件的操作

对 XML 文件的操作也是 ASP. NET 程序开发中常见的操作。下面介绍操作 XML 文件的方法。

14.3.1　XML 文件的写入

写入 XML 文件通常是其他操作的前提，因此我们先来学习 XML 文件的写入方法。下面代码的主要作用是写入 XML。

```
private void WriteXML()
```



```
{
  //创建一个表示所要生成的XML文件路径的字符串。如果该路径指向NTFS分区,则需要相关的访
    问权限
    string filename=XMLFilePathTextBox.Text;
  //创建一个写入XML数据的文件流
     System.IO.FileStream myFileStream = new System.IO.FileStream(filename, System.IO.
     FileMode.Create);
  //使用文件流对象创建一个XmlTextWriter对象
    XmlTextWriter myXmlWriter=new XmlTextWriter(myFileStream,System.Text.Encoding.Unicode);
    myXmlWriter.Formatting=Formatting.Indented;
    try
    {
      //使用WriteXMLbyXmlWriter方法把数据写入XmlTextWriter对象中
      WriteXMLbyXmlWriter(myXmlWriter,"MSFT",74.5,5.5,49020000);
      //通过Close方法的调用,XmlTextWriter对象的数据最终写入XML文件
      myXmlWriter.Close();
      Page.Response.Write("生成XML文档成功!");
    }
    catch
    {
       Page.Response.Write("生成XML文档失败!请检查路径是否正确,以及是否有写入权限。");
    }
}
private void WriteXMLbyXmlWriter(XmlWriter writer,string symbol,double price,double change,
long volume)
{
     writer.WriteStartElement("Stock");
     writer.WriteAttributeString("Symbol",symbol);
     writer.WriteElementString("Price",XmlConvert.ToString(price));
     writer.WriteElementString("Change",XmlConvert.ToString(change));
     writer.WriteElementString("Volume",XmlConvert.ToString(volume));
     writer.WriteEndElement();
}
```

14.3.2 XML 文件的读取

读取XML文件可通过三种方法来实现。

1. 通过DOM读取XML文件

```
private void ReadXMLbyDOM()
{
    //创建XmlDocument类的实例
    XmlDocument doc=new XmlDocument();
    ArrayList NodeValues=new ArrayList();
    //把people.xml文件读入内存,形成一个DOM结构
    doc.Load( Server.MapPath("people.xml") );
    XmlNode root=doc.DocumentElement;
    foreach( XmlNode personElement in root.ChildNodes )
```



```
            NodeValues.Add(personElement.FirstChild.Value);
        XMLNodeListBox.DataSource=NodeValues;
        XMLNodeListBox.DataBind();
    }
```

2. 通过 XMLReader 读取 XML 文件

（1）XmlReader 类具有一些可以在读取时修改的属性，以及一些在读取开始后被更改时并不会使新设置影响读取的其他属性。如果默认值不合适，则需要在开始读取前将这些属性设置为正确的值。但是，某些属性可以在开始读取后修改。对于不能在调用 Read 后设置的属性，读取器将引发异常。

（2）示例程序。

```
private void ReadXMLbyXmlReader()
{
    //创建 XML 读取器
    XmlTextReader reader=new XmlTextReader( Server.MapPath("students.xml") );
    ArrayList NodeValues=new ArrayList();
  while( reader.Read() )
  {
      if( reader.NodeType==XmlNodeType.Element && reader.Name=="NAME" )
      {
           reader.Read();
           NodeValues.Add( reader.Value );
      }
  }
  XMLNodeListBox.DataSource=NodeValues;
  XMLNodeListBox.DataBind();
}
```

3. 通过 XPath 读取 XML 文件

使用 XPath 来读取 XML 文件，必须引入 System.XML.XPath 命名空间。下面来看一段示例代码。

```
private void ReadXMLbyXpath()
{
    XPathDocument xpdoc=new XPathDocument( Server.MapPath("people.xml") );
    XPathNavigator nav=xpdoc.CreateNavigator();
    XPathExpression expr=nav.Compile("descendant::PEOPLE/PERSON");
    XPathNodeIterator iterator=nav.Select(expr);
    ArrayList NodeValues=new ArrayList();
    while (iterator.MoveNext())
        NodeValues.Add(iterator.Current.ToString());
    XMLNodeListBox.DataSource=NodeValues;
    XMLNodeListBox.DataBind();
}
```

14.3.3　XML 文件的显示和验证



1. XML 文件的显示

```
//通过 XSL 显示 XML 文件
private void DisplayXML()
{
    System.Xml.XmlDocument xmldoc=new System.Xml.XmlDocument();
    xmldoc.Load(Server.MapPath("user.xml"));
    System.Xml.Xsl.XslTransform xmltrans=new System.Xml.Xsl.XslTransform();
    xmltrans.Load(Server.MapPath("user.xsl"));
    Xml1.Document=xmldoc;
    Xml1.Transform=xmltrans;
}
```

2. XML 文件的验证

```
private void ValidateXML()
{
    FileStream stream=new FileStream(Server.MapPath("people.xml"),FileMode.Open);
    //创建 XmlValidatingReader 类的对象
    XmlValidatingReader vr=new XmlValidatingReader(stream,XmlNodeType.Element,null);
    //加载 XML 架构文档
    vr.Schemas.Add(null,Server.MapPath("people.xsd"));
    //说明验证的方式是根据 XML 架构
    vr.ValidationType=ValidationType.Schema;
    vr.ValidationEventHandler+=new ValidationEventHandler(ValidationHandler);
    //对文档进行验证
    while(vr.Read());
    //显示验证过程完成
    Page.Response.Write("<b>Validation finished! <b>");
    //关闭打开的文件
    stream.Close();
}
private void ValidationHandler(object sender,ValidationEventArgs args)
{
    //显示验证失败的消息
    Page.Response.Write("<b>Validation error:</b>"+args.Message+"<p>");
}
```

练　习

1. 单项选择题

(1) 文件上传通常由________控件来完成。

A. FileUpload　　B. TextBox　　C. Button　　D. Label

(2) FileUpload 控件的________属性用来表示上传文件的名称。

A. FileName　　B. Name　　C. FileContent　　D. ContentType

(3) 要实现对文件进行操作必须引用________命名空间。



A. System.IO　B. System.Data　C. System.Web　D. System.Array

(4) 用来创建目录的方法是________。

A. Create　B. Move　C. CreateDirectory　D. Get

(5) 读取 XML 文件的方法不包括________。

A. 通过 DOM 读取 XML 文件　B. 通过 XMLReader 读取 XML 文件

C. 通过 DataReader 读取 XML 文件　D. 通过 XPath 读取 XML 文件

2. 简答题

(1) 简述文件上传和下载的主要方法。

(2) 简述操作文件的 File 类的主要方法。

(3) 简述操作目录的 Directory 类的主要方法。

实　训

实训目的

(1) 熟练掌握使用 Upload 控件上传和下载文件的方法。

(2) 熟悉文件和目录的操作方法。

实训内容

(1) 启动 Visual Studio 2005。

(2) 打开 Visual Studio 开发环境中解决方案资源管理器、工具箱、属性窗口和设计视图。

(3) 新建一个网站 Exercise 14,实现一个文件上传和下载的程序,单击"上传"按钮,可以上传到指定文件夹,单击"下载"按钮可以将文件下载下来。

(4) 将上面的网站发布。

(5) 配置 IIS,在 IIS 服务器上文件上传页面。

任务15　网站部署与定制

技能目标

会使用站点管理工具配置站点,能熟练地对在线考试系统进行系统配置。

知识目标

了解 ASP.NET 的各种配置文件,掌握 ASP.NET 的常用配置。

任务描述

网站开发好之后,需要进行迁移和发布。之前,通常都会对网站进行部署,如设置网站的个性化信息,设置网站的安全性等。

本章任务如下:

◆ 学习 ASP.NET 配置文件的作用、类型结构;

◆ 学习网站管理工具的使用方法;

◆ 配置在线考试系统的数据库连接,进行身份验证。

15.1　知识准备

15.1.1　ASP.NET 中的配置文件概述

在许多应用程序中,需要存储并使用标识用户唯一的信息。当该用户访问站点时,就可以使用已存储的信息向用户显示 Web 应用程序的个性化版本。个性化应用程序需要大量的元素,必须使用唯一的用户标识符存储信息,能够在用户再次访问时识别用户,然后根据需要获取用户信息。若要简化应用程序,可以使用 ASP.NET 配置文件功能,该功能可以实现所有上述任务。

ASP.NET 配置文件功能将信息与单个用户关联,并采用持久性的格式存储这些信息。通过配置文件,可以管理用户信息,而无须创建和维护自己的数据库。此外,通过使用可从应用程序中的任何位置访问的强类型 API,就可以借助 ASP.NET 配置文件功能使用相关的用户信息。

可以使用配置文件存储任何类型的对象。配置文件功能提供了一项通用存储功能,能够定义和维护几乎任何类型的数据,同时仍可用类型安全的方式使用数据。

配置文件的特点如下:



- 基于 XML 文本文件，它用来储存 ASP. NET Web 应用程序的配置信息；
- 可以出现在应用程序的每一个目录中，所有的子目录都继承它的配置设置；
- 在运行时对配置文件的修改不需要重启服务就可以生效；
- 可以自定义新配置参数并编写配置节处理程序以对它们进行处理。

15.1.2　ASP. NET 中的配置文件类型

在 ASP. NET 中，配置文件主要有两种，这两种配置文件也形成了配置文件的层次结构。

1. machine. config

machine. config 是最高级别的配置文件，它可以实现对整个计算机所有应用的配置。通常，在 ASP. NET 中，很多默认的配置都是定义在这个配置文件中的，但是为了缩短 ASP. NET 的加载时间，有一些配置已经从 machine. config 中移出到别的地方了。

2. Web. config

Web. config 是最常用的配置文件，它可以存放在一个网站的根目录中，以对整个根目录起作用，这是最常见的配置地方。此外，还可以把 Web. config 文件放置在网站中的某个文件夹中的 Web. config 文件中，此时可以通过这个配置文件来限制对文件的访问权限。

ASP. NET 配置文件的层次结构如图 15-1 所示。

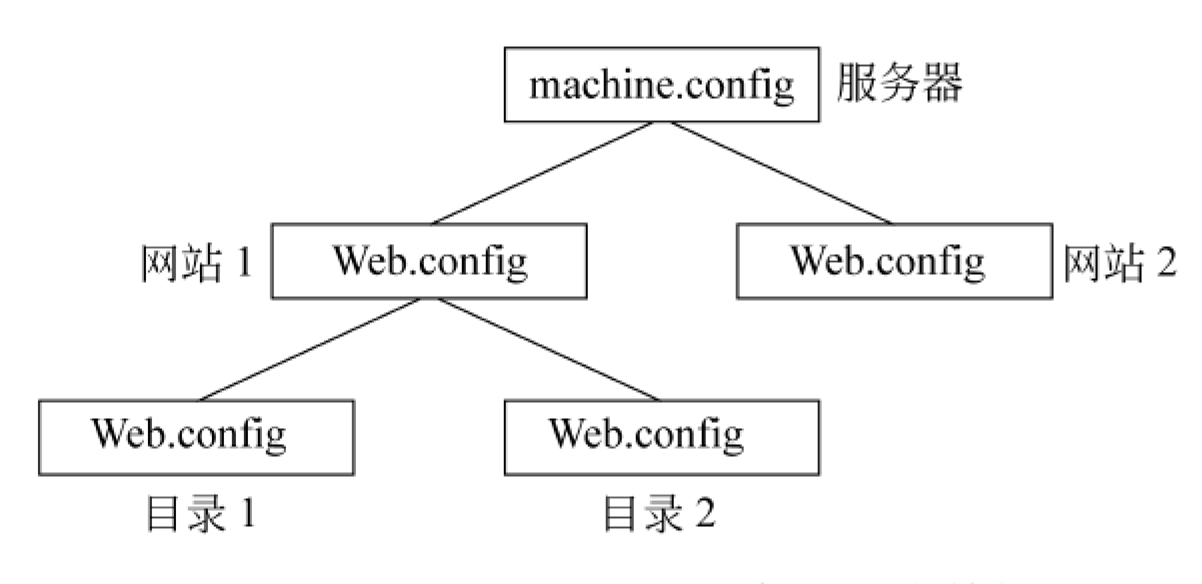


图 15-1　ASP. NET 配置文件的层次结构

15.1.3　ASP. NET 中的配置文件结构

1. XML 标记的命名

ASP. NET 配置文件是基于 XML 的文本文件，每个配置文件都包含 XML 标记(配置节)和子标记的嵌套层次结构，这些标记带有指定配置设置的属性。因为这些标记必须是格式正确的 XML，所以标记、子标记和属性是区分大小写的。标记名和属性名是 Camel 大小写形式的，即标记名的第一个字符是小写的，任何后面连接单词的第一个字母是大写的。属性值是 Pascal 大小写形式的，即第一个字符是大写的，任何后面连接单词的第一个字母也是大写的；但是 true 和 false 例外，它们总是小写的。



2. Web.config 配置文件的结构

因为 Web.config 是常用的配置文件，所以这里着重讲解 Web.config 配置文件的结构。在 Visual Studio 2005 中不会自动生成 Web.config 文件，建立的方法是：在已经建立好的网站上右击，从弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框。在此对话框中，选择“Web 配置文件”选项，如图 15-2 所示，单击“添加”按钮，即可创建一个 Web.config 文件。



图 15-2　添加 Web 配置文件

所有配置信息都驻留在 <configuration> 和 </configuration> 根 XML 标记之间。标记间的配置信息分为两个主区域：配置节处理程序声明区域和配置节设置区域。

配置节处理程序声明出现在配置文件顶部 <configSections> 和 </configSections> 标记之间。包含在 <section> 标记中的每个声明都指定提供特定配置数据集的节的名称和处理该节中配置数据的.NET Framework 类的名称。

配置节设置区域位于 <configSections> 区域之后，它包含实际的配置设置。<configSections> 区域中的每个声明都有一个配置节。每个配置节都包含子标记，这些子标记带有包含该节设置的属性。

下面介绍常用的配置节的作用和示例。

1）<appSettings>节

（1）作用：用来存储 ASP.NET 应用程序的一些配置信息，比如上传文件的保存路径等。

（2）示例

存储上传文件的信息：

```
<appSettings>
        <!--上传文件夹位置-->
        <add key="FileUplodePath" value="Upload"/>
        <!--上传文件类型-->
        <add key="FileTypeLimit" value=".zip,.rar,.jpg,.gif,.bmp,.doc"/>
        <!--限制单个文件大小,单位为 KB,10240KB=10MB-->
```



```
        <add key="FileSizeLimit" value="10240"/>
        <!--限制文件夹的总大小,单位为 KB,102400KB=100MB-->
        <add key="FolderSizeLimit" value="102400"/>
</appSettings>
```

2) <connectionStrings>节

(1) 作用：用于配置数据库连接，可以在<connectionStrings>节中增加任意一个节点来保存数据库连接字符串，将来通过代码的方式动态获取节点的值来实例化数据库连接对象。这样一旦部署的时候数据库连接信息发生变化，仅需要更改此处的配置即可，不必因为数据库连接信息的变化而需要改动程序代码和重新部署。

(2) 示例

```
<connectionStrings>
  <add name="ConnectionString1" connectionString="Data Source=(local);Initial
           Catalog=School;User ID=sa;Password=123"/>
</connectionStrings>
```

3) <compilation>节

作用：配置 ASP.NET 使用的所有编译设置。默认的 debug 属性为 true，在程序编译完成交付使用之后应将其设为 true。

4) <authentication> 节

(1) 作用：配置 ASP.NET 身份验证支持，身份验证方式分为 Windows、Forms、PassPort、None 四种，其中 Forms 验证最为常用。该元素只能在计算机、站点或应用程序级别声明。<authentication>节通常与<authorization>节配合使用。

(2) 示例

以下示例为基于窗体(Forms)的身份验证配置站点，当没有登录的用户访问需要身份验证的网页，网页自动跳转到登录网页。

```
<authentication mode="Forms" >
    <forms loginUrl="login.aspx" name="FormsAuthCookie"/>
</authentication>
```

其中，

✍ loginUrl：表示登录网页的名称。

✍ name：表示 Cookie 名称。

5) <authorization> 节

(1) 作用：控制对 URL 资源的客户端访问(如允许匿名用户访问)。此元素可以在任何级别(计算机、站点、应用程序、子目录或页)上声明，必须与<authentication> 节配合使用。

(2) 示例

以下示例允许 admin 用户访问，禁止匿名用户的访问。

```
<authorization>
  <allow roles="admin">
  <deny users="?"/>
</authorization>
```



其中，

- allow 表示允许。
- deny 表示拒绝。
- 特殊符号："＊"代表所有用户，"?"代表匿名用户。
- 授权的配置顺序非常重要，系统总是按照从前向后逐条匹配的方式，执行最先的匹配者。

小贴士

可以使用 Web. Security. FormsAuthentication. RedirectFromLoginPage 方法将已验证的用户重定向到用户刚才请求的页面。

6）<customErrors>节

（1）作用：为 ASP. NET 应用程序提供有关自定义错误的信息。但它不适用于 XML Web services 中发生的错误。

（2）示例

当发生错误时，将网页跳转到自定义的错误页面。

```
<customErrors defaultRedirect="Error.aspx" mode="RemoteOnly">
</customErrors>
```

其中，

- defaultRedirect：表示自定义的错误网页的名称。
- mode：显示自定义错误的模式。On 表示启用自定义错误；Off 表示禁用自定义错误；RemoteOnly 表示本地用户将看到详细错误信息，而远程用户将会看到自定义错误信息。

7）<httpRuntime>节

（1）作用：httpRuntime 是配置 ASP. NET HTTP 运行时的设置，以确定如何处理对 ASP. NET 应用程序的请求。该节可以在计算机、站点、应用程序和子目录级别声明。

（2）示例

控制用户上传文件最大为 4MB，最长时间为 60 秒，最多请求数为 100。

```
<httpRuntime maxRequestLength="40960" executionTimeout="60"
             appRequestQueueLimit="100"/>
```

其中，

- maxRequestLength：控制最大上传的文件大小，单位为 KB。该限制可用于防止因用户将大量文件传递到该服务器而导致的拒绝服务攻击。默认值为 4096KB（4MB）。
- executionTimeout：表示允许执行请求的最大时间限制，单位为秒。
- appRequestQueueLimit：表示 ASP. NET 将为应用程序排队的请求的最大数目。当没有足够的自由线程来处理请求时，将对请求进行排队。当队列超出了该设置中指定的限制时，将通过"503-服务器太忙"错误信息拒绝传入的请求。

8）<pages>节

（1）作用：标识特定针对于页的配置设置（如是否启用会话状态、视图状态，是否检测用户的输入等）。<pages>可以在计算机、站点、应用程序和子目录级别声明。



（2）示例

不检测用户在浏览器输入的内容中是否存在潜在的危险数据，在从客户端向服务器发送页面时将检查加密的视图状态，以验证视图状态是否已在客户端被篡改。

```
<pages buffer="true" enableViewStateMac="true" validateRequest="false"/>
```

其中，

- buffer：是否启用了 HTTP 响应缓冲。
- enableViewStateMac：是否应该对页的视图状态运行计算机身份验证检查（MAC），以放置用户篡改。默认为 false，如果设置为 true 将会引起性能的降低。
- validateRequest：是否验证用户输入中有跨站点脚本攻击和 SQL 注入式漏洞攻击，默认为 true，如果出现匹配情况就会发 HttpRequestValidationException 异常。对于包含有在线文本编辑器页面一般自行验证用户输入，而将此属性设为 false。

9）<sessionState>节

（1）作用：为当前应用程序配置会话状态设置（如设置是否启用会话状态，会话状态保存位置）。

（2）示例

```
<sessionState mode="InProc"cookieless="true"timeout="20"/>
</sessionState>
```

其中，

- mode：表示在本地储存会话状态的模式。
- cookieless：表示如果用户浏览器不支持 Cookie 时是否启用会话状态（默认为 false）。
- timeout：表示会话可以处于空闲状态的分钟数。

10）<trace>节

（1）作用：配置 ASP. NET 跟踪服务，主要用来进行程序测试，判断哪里出错。

（2）示例

以下为 Web. config 中的默认配置。

```
<trace enabled="false" requestLimit="20" pageOutput="false" traceMode=
"SortByTime" localOnly="true" />
```

其中，

- enabled：表示是否启用跟踪。
- requestLimit：表示指定在服务器上存储的跟踪请求的数目。
- pageOutput：表示只能通过跟踪实用工具访问跟踪输出。
- traceMode：表示 trace 内容排序的模式，"SortByTime"表示以处理跟踪的顺序来显示跟踪信息。
- localOnly：表示跟踪查看器（trace. axd）是否只用于宿主 Web 服务器。

15.1.4 网站管理工具

对网站的配置除了可以使用手动配置 Web. config 文件的方法实现之外，还可以通过网



站管理工具来实现。网站管理工具使网站开发人员能够通过简单的 Web 界面查看并管理网站配置。

如果要使用网站管理工具来实现对网站的配置管理，必须先打开网站管理工具。打开网站管理工具的方法如下：

（1）利用 Visual Studio 2005 IDE 的“网站”菜单下的“ASP. NET 配置”命令来打开，如图 15-3 所示。

（2）利用解决方案资源管理器的工具栏中的“ASP. NET 配置”图标来打开，如图 15-4 所示。



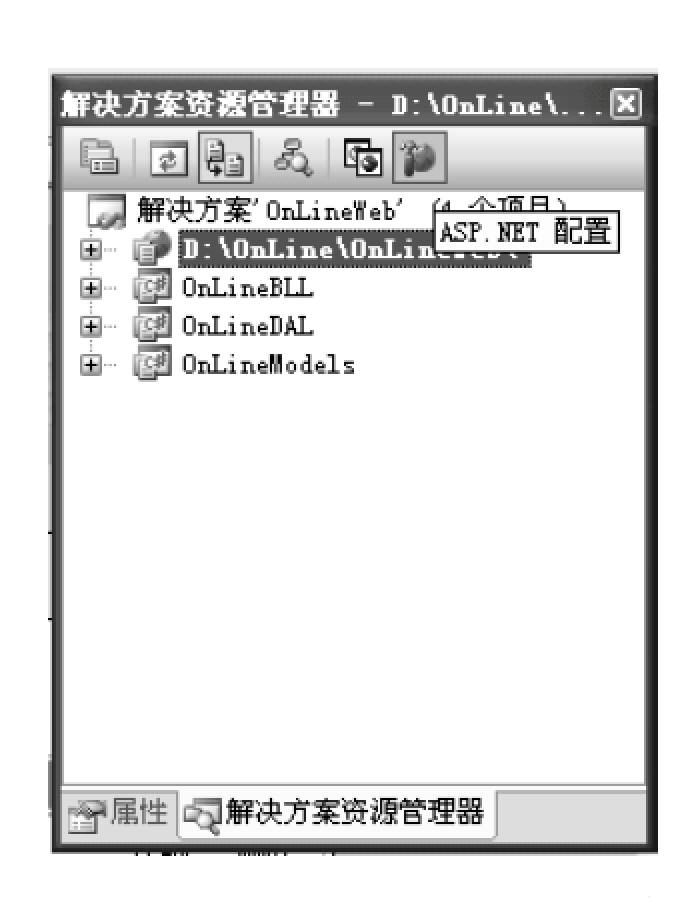



图 15-3 “网站”菜单下的“ASP. NET 配置”命令

图 15-4 “ASP. NET 配置”图标

打开网站管理工具之后，即可进入网站管理工具主页，如图 15-5 所示。

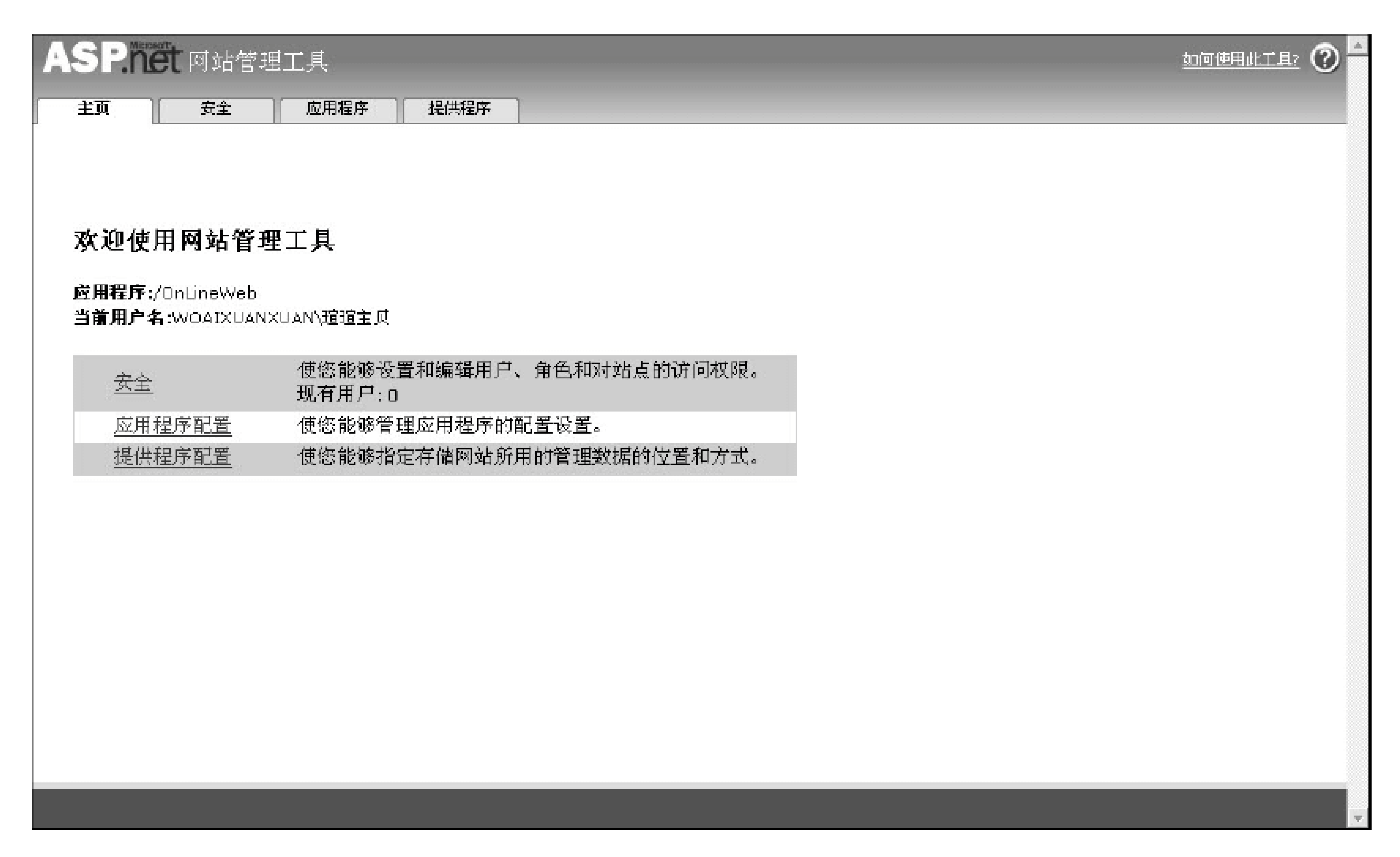


图 15-5 网站管理工具主页

网站管理工具的主页中给出了需要进行配置的应用程序名称、当前用户名，主页中有 3 个链接选项，可供链接到 3 个配置页面。此外，通过“主页”选项卡旁边的 3 个选项卡，也可以对网站进行 3 个方面的配置，分别是：“安全”配置、“应用程序”配置和“提供程序”配



置。下面分别对这 3 方面的配置作一简单介绍。

小贴士

使用 ASP.NET 网站管理工具对网站进行安全配置可能会修改 Web.config，因此在打开配置前应先关闭 Web.config 文件。

1. “安全”配置

单击“安全”标签或“安全”链接，可以进入“安全”配置页面，如图 15-6 所示。在此页面中，可以用网站管理工具来管理应用程序的所有安全设置，包括：用户管理（成员资格管理）、角色管理、访问规则管理（授权）。

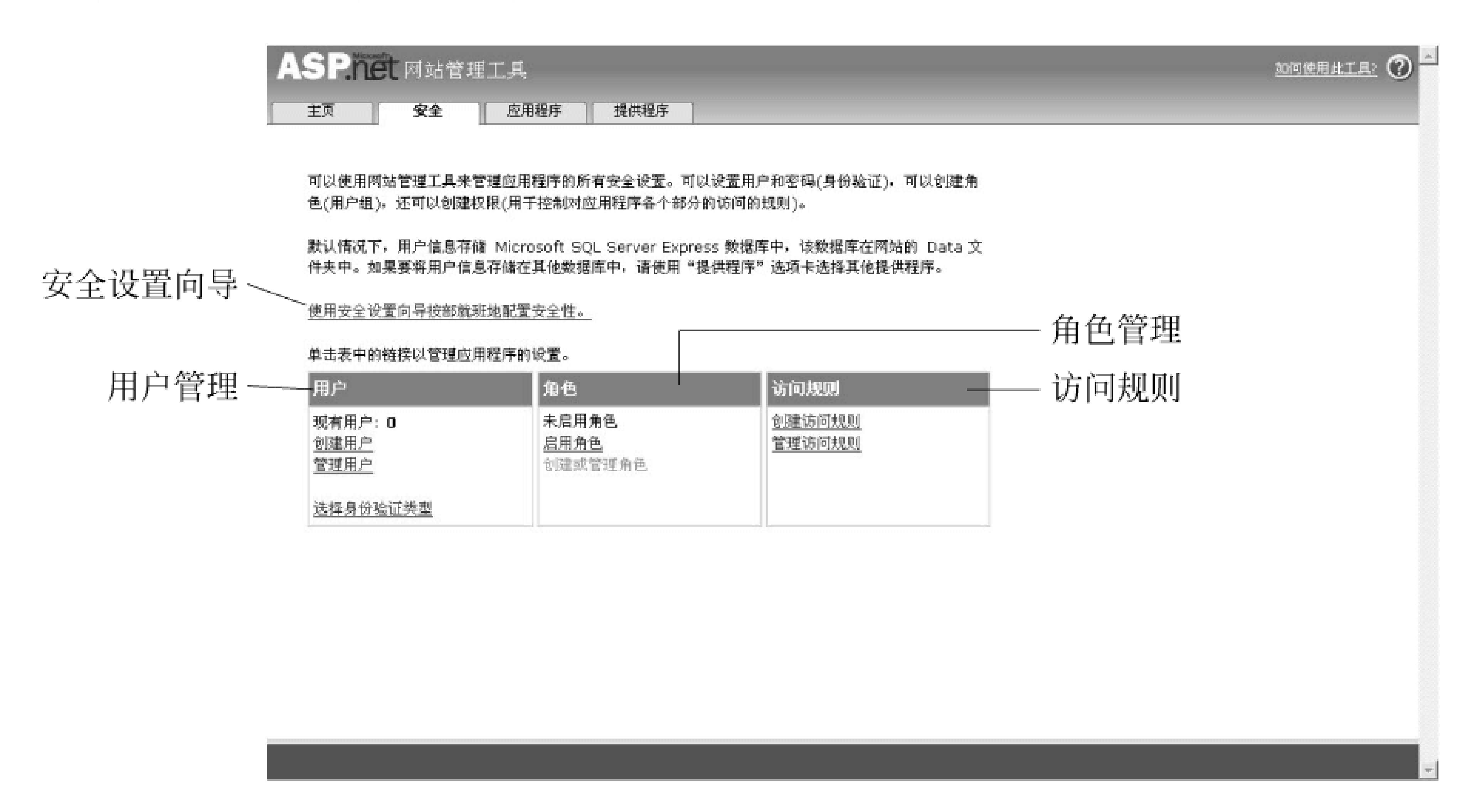


图 15-6　“安全”配置页面

（1）用户管理

用户管理功能仅对 Forms 验证有效，通常验证方式为默认的基于 Windows 的身份验证，如果需要对用户进行管理，需要单击“用户”选项区域中的“选择身份验证类型”链接，然后选择“通过 Internet”选项，就可以看到用户管理的一些功能了，如图 15-7 所示。

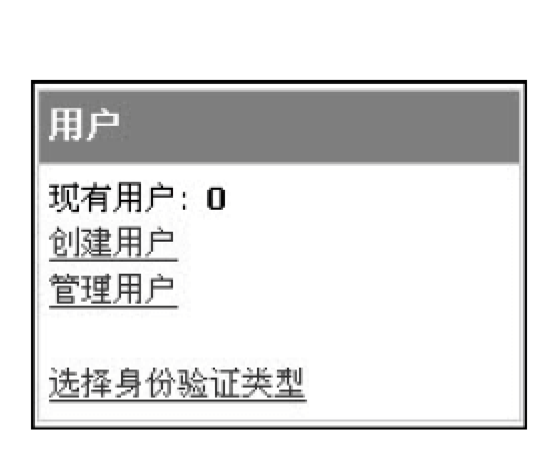


图 15-7　用户管理

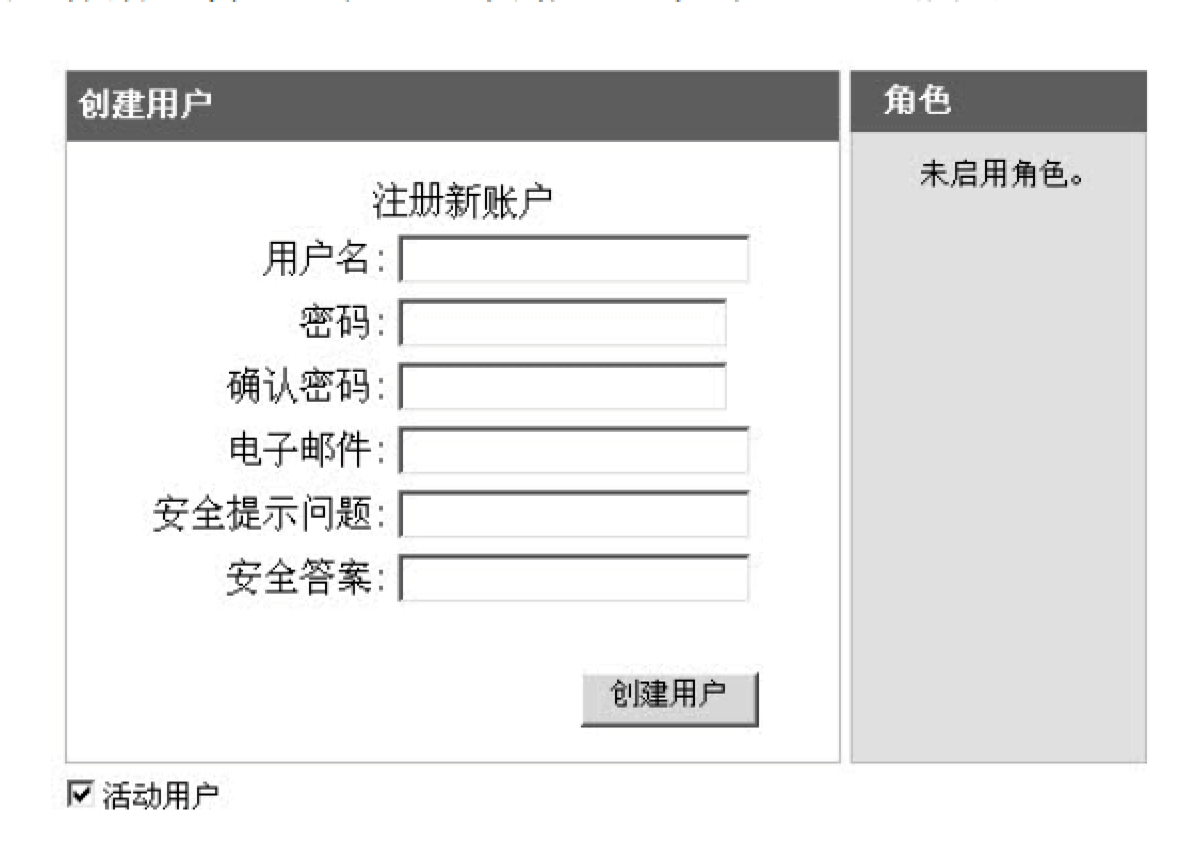


图 15-8　注册新账户



在“用户”选项区域中，单击“创建用户”链接可以创建用户账号，如图 15-8 所示。单击“管理用户”链接可以对已有的一些用户进行管理操作：搜索、编辑、删除和配置角色，如图 15-9 所示。

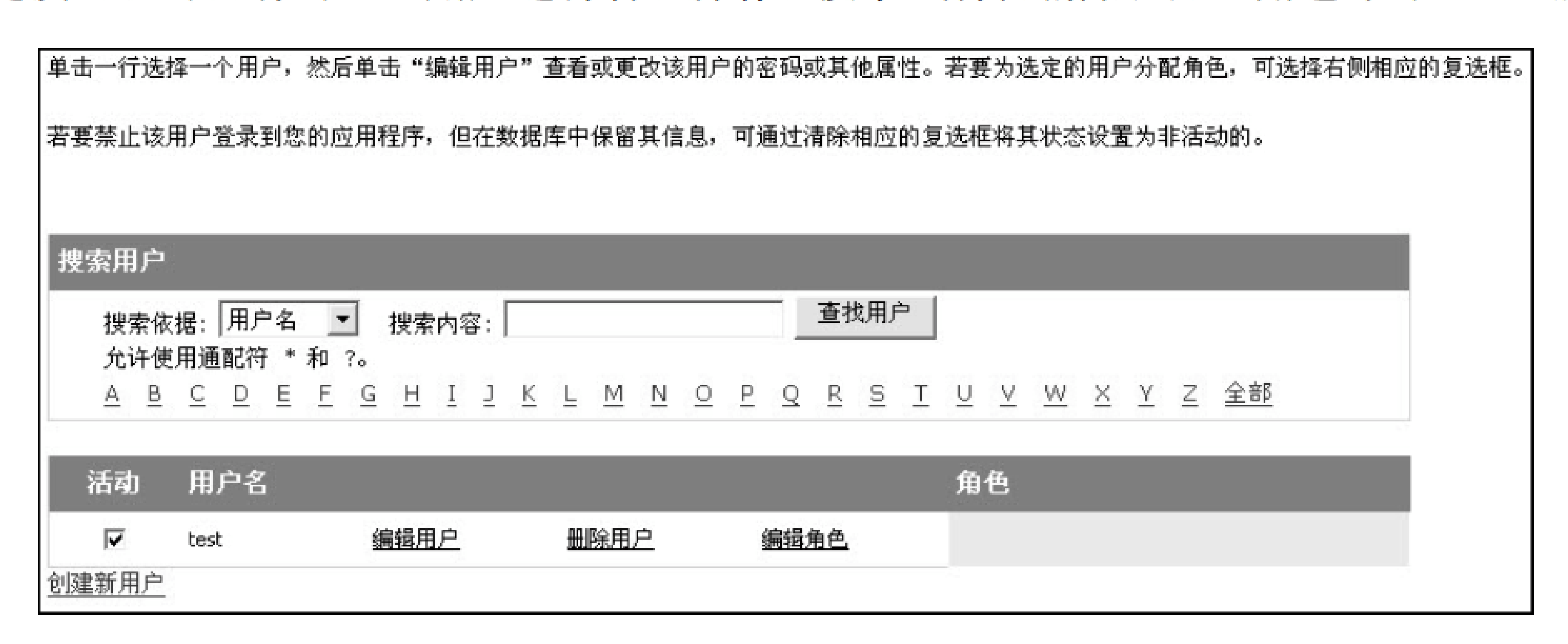


图 15-9　管理用户界面

创建好用户之后，用户信息都被保存在 ASPNETDB 数据库中。此时还未启用角色，因此暂时没有配置角色的功能。

（2）角色管理

如果没有启用角色管理，将会看到如图 15-10 所示的界面，单击“启用角色”链接后界面的效果如图 15-11 所示。角色启用之后，会在 Web. config 的＜system. Web＞节点下看到如下配置：＜roleManager enabled="true" /＞。



图 15-10　未启用角色管理

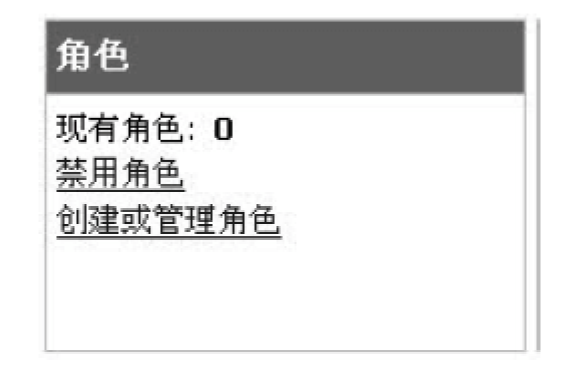


图 15-11　已经启用角色管理

单击“创建或管理角色”链接可以进行角色的创建、修改和删除，如图 15-12 所示。

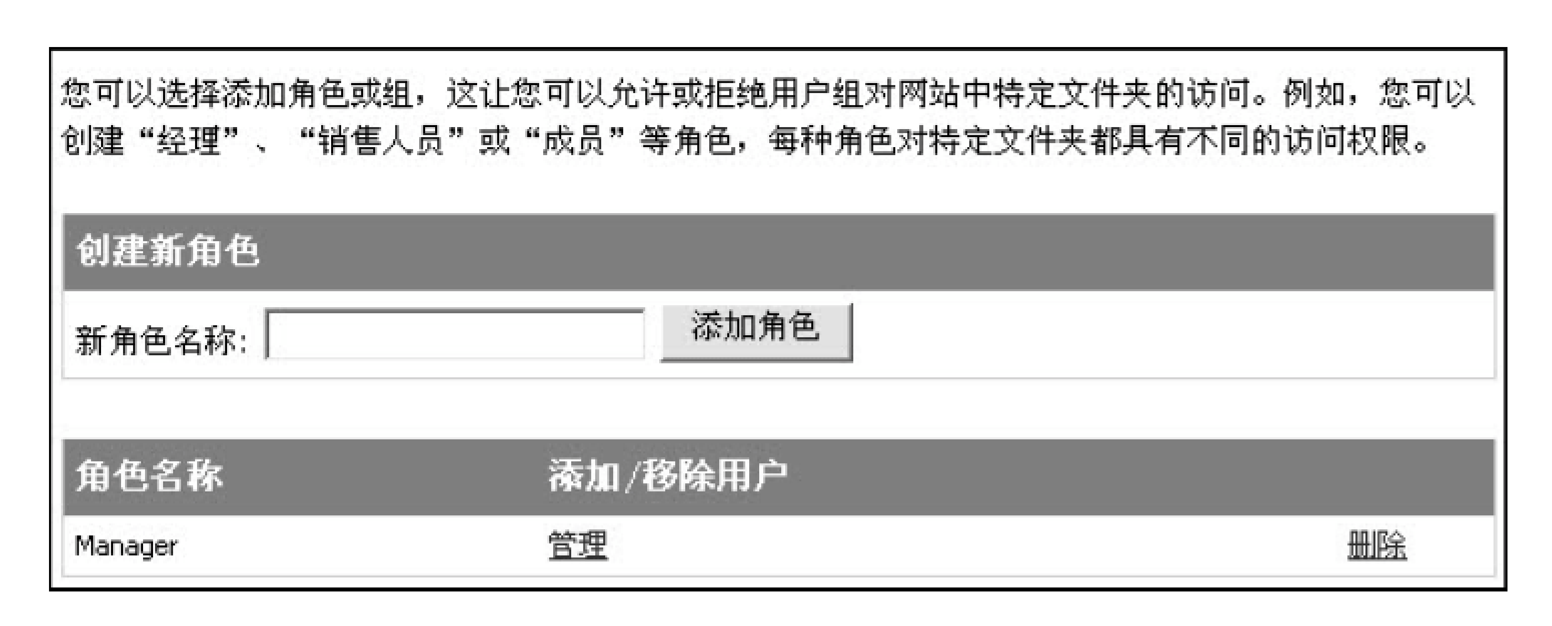


图 15-12　角色管理

在这里，我们新建了一个 Manager 角色，但并没有把任何用户关联到这个角色，也没有对这个角色进行授权。现在使用用户管理功能创建一个名为 SalesManager 的用户，并在创建的时候直接将其关联到 Manager 这个角色，如图 15-13 所示。

（3）访问规则管理

用户和角色创建好之后，可以建立相应的访问规则，来规定对网站或目录的访问权限。





创建用户　角色

注册新账户

用户名：testld

密码：●●●●●●

确认密码：●●●●●●

电子邮件：test@126.com

安全提示问题：haha

安全答案：hehe

请另输入一个用户名。

创建用户

为此用户选择角色：

Manager

活动用户

图 15-13　创建用户并关联角色

单击“创建访问规则”链接，即可进入如图 15-14 所示的页面。

您可以选择添加访问规则来控制对整个网站或单个文件夹的访问。规则可应用于特定的用户和角色、所有用户、匿名用户或这些用户的某种组合。规则应用于子文件夹。

添加新访问规则

为此规则选择一个目录：　OnLineWeb　admin　App_Data　Bin　CSS　image　_notes

规则应用于：　角色 Manager　用户　搜索用户　所有用户　匿名用户

权限：　允许　拒绝

图 15-14　创建访问规则

2．“应用程序”配置

在“应用程序”配置页面中，系统提供了许多与应用程序相关的配置，包括应用程序设置、SMTP 邮件服务器设置、调试和跟踪设置、整个应用程序的启动和调试以及定义默认错误页，如图 15-15 所示。



图 15-15　“应用程序”配置页面



3. “提供程序”配置

利用它可配置网站管理数据的存储方式，可以对站点管理只使用一个提供程序，也可以为每种功能指定不同的提供程序，“提供程序”配置页面如图 15-16 所示。



图 15-16　“提供程序”配置页面

15.2　任务实施

15.2.1　数据库连接配置

1. 配置数据库连接字符串

在 ASP. NET 2.0 中通常可以将数据库连接字符串保存于.aspx 页面中，为了保持一致性及提高效率，也可以将 ConnectionString 属性保存于 Web. config 文件中。通常，数据库连接字符串存储在 Web. config 文件中的<connectionStrings>和</connectionStrings>的 XML 标记之间。

打开在线考试系统，在解决方案资源管理器中，右击“D:/Online/OnlineWeb”，从弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，如图 15-17 所示。在“Visual Studio 已安装的模板”选项区域中选择“Web 配置文件”选项，在“名称”文本框中输入“Web. config”，单击“添加”按钮。

这样，系统生成 Web. config 配置文件，在解决方案管理器中双击将其打开。向里面添加内容如下：

```
<connectionStrings>
    <add name="onLineExam"connectionString="DataSource=.;Initial Catalog=onLineExam1;
      User ID=sa;password=123"/>
</connectionStrings>
```



图 15-17　“添加新项”对话框

小贴士

<connectionStrings>配置节可以通过手动方式添加到 Web. config 配置文件中，并且可以通过添加多个 add 标签，来保存多个数据库连接的配置。连接字符串配置完成之后，如果数据库的位置发生改变，直接修改配置文件中的连接信息即可。

2. 使用连接配置

连接配置完成之后，可以在 ConnDBHelper. cs 文件中使用该连接字符串。使用的方法如下：

```
string connectionString = ConfigurationManager. ConnectionStrings [ " onLineExam "].
ConnectionString;
```

如果在使用的时候，发现 ConfigurationManager 无法识别，则说明该类文件缺少 System. Configuration 命名空间。可以先为 OnLineDAL 添加引用，将 System. Configuration 导入，然后再给该类文件添加 using System. Configuration 即可。

小贴士

在 ASP. NET 1. 1 中创建连接字符串的方式如下：

```
<appSettings>
    <add key="con"value="server=a;database=northwind;uid=sa;pwd=123"/>
</appSettings>
```

可以通过使用 ConfigurationSettings. AppSettings 静态字符串集合来访问 Web. config 文件。

示例：获取上面例子中建立的连接字符串。

```
String con=ConfigurationSettings.AppSettings["SconStr"]
```



15.2.2　身份验证配置

在线考试系统分为学生考试模块和管理模块，对于管理模块，只有管理员用户才能访问，即要设置 admin 文件夹的访问权限，此时必须通过身份验证配置才能实现。对 admin 文件夹的访问权限的设置，分为以下三个步骤实现。

（1）配置网站根目录的 Web. config

打开 Web. config 文件，修改<authentication>配置节，这里需要设置 mode 为 Forms 验证方式，Forms 验证的 name 为 login，具体路径 loginUrl 为"～/admin/AdminLogin. aspx"，失效时间为 timeout＝"60"。具体代码如下：

```
<authentication mode="Forms">
    <forms name="Login" loginUrl="~ /admin/AdminLogin.aspx" timeout="60"/>
</authentication>
```

（2）配置 admin 目录的 Web. config

为了给 admin 目录设置访问权限，还需要在 admin 目录下添加 Web. config，然后对它进行权限设置。设置的方法是配置<authorization>配置节，这里设置拒绝匿名用户访问。具体代码如下：

```
<configuration>
    <appSettings/>
    <connectionStrings/>
    <system.Web>
      <authorization>
        <deny users="?"/>
      </authorization>
    </system.Web>
</configuration>
```

（3）更改登录代码

上述两步配置完成之后，还需要修改 AdminLogin. aspx 页面的“进入系统后台”按钮的代码，以使 Forms 验证起作用。具体代码如下：

```
protected void btnAdminLogin_Click(object sender,EventArgs e)
{
  if (AdminManager.AdminLogin(this.txtName.Text,this.txtPwd.Text))
 {
  Session["LoginUser"]=this.txtName.Text;
  string strRedirect;
  strRedirect=Request["ReturnUrl"];
  System.Web.Security.FormsAuthentication.SetAuthCookie(this.txtName.Text,true);
  if (strRedirect==null)
     Response.Redirect("~ /admin/ListAllStudents.aspx");
 Response.Redirect(strRedirect);
 }
 else
 {
```



```
15        Response.Write("<script>alert('用户名或密码错误!');</script> ");
16    }
17  }
```

代码剖析：

第 3 行表示调用 OnLineBLL 业务逻辑层的 AdminManager 类中的 AdminLogin 方法来判定是否存在该用户；

第 5 行表示设置 Session 中 LoginUser 的值与 txtName 中的值相同；

第 6 行表示定义表单验证所指定的路径；

第 7 行表示读取 Request["ReturnUrl"]并赋值给 strRedirect；

第 8 行表示执行表单验证；

第 9 行表示判定 strRedirect 路径是否为空；

第 10 行表示路径为空时，重定向到 admin 文件夹下的 ListAllStudents. aspx 页面；

第 11 行表示路径不为空时，重定向到 strRedirect 指定的页面；

第 15 行表示用户不存在时，提示“用户名或密码错误!”。

至此，对 admin 文件夹的访问权限设置成功，当匿名用户访问的时候，系统会自动转入 AdminLogin. axpx 页面进行用户登录。如果登录成功，系统会自动跳转到初始访问页面；如果登录失败，系统会提示“用户名或密码错误!”。

15.2.3 自定义错误

在 Web 应用程序运行期间，应用程序也许会遇到错误而抛出异常信息，此时我们更希望用户看到友好的用户界面而非单纯的错误代码。ASP. NET 提供的自定义错误机制可以使我们自己定义错误，从而使错误发生时，程序自动跳到发生错误的界面。

在 ASP. NET 中自定义错误的方法主要有两种，下面分别加以介绍。

方法 1

(1) 配置 Web. config

首先配置网站根目录下的 Web. config，方法如下：

```
<system.Web>
    <customErrors mode="On" defaultRedirect="Error.aspx" >< /customErrors>
</system.Web>
```

(2) 添加错误处理页面 Error. aspx，调用下面的方法。

```
private void DealError()
{
    HttpException error=new HttpException();
    string strCode=error.ErrorCode.ToString();
    string strMsg=error.Message;
    error.HelpLink="sss";
    Response.Write("ErrorCode:"+strCode+"<br>");
    Response.Write("Message:"+strMsg+"<br>");
```



```
    Response.Write("HelpLink:"+error.HelpLink+"<br>");
    Response.Write("Source:"+error.Source+"<br>");
    Response.Write("TargetSite:"+error.TargetSite+"<br>");
    Response.Write("InnerException:"+error.InnerException+"<br>");
    Response.Write("StackTrace:"+error.StackTrace+"<br>");
    Response.Write("GetHtmlErrorMessage:"+error.GetHtmlErrorMessage()+"<br>");
    Response.Write("error.GetHttpCode().ToString():"+error.GetHttpCode().ToString ()+"<br>");
    Response.Write("error.Data.ToString()::"+error.Data.ToString()+"<br>");
}
```

这种方法不能完整地显示错误信息。

方法 2

(1) 配置 Web. config

首先配置网站根目录下的 Web. config,方法如下:

```
<system.Web>
    <customErrors mode="On" defaultRedirect="ApplicationError.aspx">
    </customErrors>
</system.Web>
```

(2) 在 Global. asax 文件中找到 Application_Error 事件,加入代码如下:

```
Exception error=Server.GetLastError();
string err="出错页面是:"+Request.Url.ToString()+"<br> ";
err+="异常信息:"+error.Message+"<br> ";
err+="Source:"+error.Source+"<br> ";
err+="StackTrace:"+error.StackTrace+"<br> ";
Server.ClearError();
Application["error"]=err;
```

(3) 添加错误处理页面

在 Error. aspx 中加入代码如下:

```
Response.Write(Application["error"].ToString());
```

这种方法能完整地显示错误信息。

15.2.4 sessionState 配置

Web 应用程序的网页是基于 HTTP 的,它们没有状态标识存储,即它们不知道所有的请求是否来自同一台客户端计算机,网页是否受到了破坏,以及是否得到了刷新,这样就可能造成信息的丢失。于是状态管理就成了开发网络应用程序的一个实实在在的问题。

在 ASP 中能够通过 Cookie、查询字符串、应用程序、会话(Session) 等轻易解决这些问题。现在在 ASP. NET 环境中,依然可以使用这些功能,并且功能更加强大。

状态管理分为服务端和客户端两种情况,这里只介绍服务端状态管理。

与 Application 对象不同的是,ASP. NET 的 Session 对象可以在 IIS 服务器或者工作



进程重新启动时恢复启动前的状态而不丢失其中的数据。这是因为存储在 Session 中的所有信息都默认存储在一个作为 Windows 服务运行的状态服务器进程中。状态可以被序列化并以二进制形式保存在内存中。程序员可以选择使用 Microsoft SQL Server 数据库来存储数据。

状态服务器服务和状态信息可以和 Web 应用程序一起存在于同一台服务器上，也可以保存到外部的状态服务器上。为了指定如何存储信息，程序员可以在 Web. config 文件中编写适当的配置。

ASP. NET 会话状态模块在 Web. config 文件中<System. Web>标记下的<Sessionstate>标记的 mode 属性来决定该属性的 4 种可能的值：InProc、Off、StateServer 和 SQL Server。下面分别对这 4 个值的含义作一介绍。

1. InProc

InProc 是默认设置，它允许“无 Cookie”的会话，以及在服务器之外存储会话数据。ASP. NET 会话状态模块在 Web. config 文件中配置如下：

```
<sessionState mode="InProc"cookieless="false" timeout="20" />
```

在这个例子中，mode 属性设为 InProc（默认值），表明会话状态要由 ASP. NET 存储到内存中，而且不用 Cookie 来传递会话 ID。相反，会话 ID 要直接插入一个网页 URL 的查询字符串中。例如，采用 InProc 模式并建立一个会话之后，调用一个假想的 ASP. NET 网页时，需要采用下面这样的 URL。

```
http://my.Website.com/(12mfju55vgblubjlwsi4dgjq)/education.aspx
```

圆括号中长长的字母、数字字符串就是会话 ID。ASP. NET 引擎从查询字符中提取会话 ID，并将用户请求与特定会话联系起来。采取这种方式，不管 Cookie 还是隐藏表单字段都用不着了。所以，即使网页中没有使用表单，也能加入会话。

但是这种方法，应用程序的状态将依赖于 ASP. NET 进程，当 IIS 进程崩溃或者正常重启时，保存在进程中的状态将丢失。

2. Off

和从前的 ASP 一样，ASP. NET 的会话状态管理是要产生开销的。所以，假如某个网页不需要访问 Session 对象，开发者应将那个页面的 Page 预编译指令的 EnableSessionState 属性设为 false。要为整个网站禁用会话状态，可在 Web. config 文件中将 sessionState 元素的 mode 属性设为 Off。

为了克服 InProc 模式的缺点，ASP. NET 提供了两种进程外保存会话状态的方法。

3. StateServer

将 mode 属性设为 StateServer，也就是将会话数据存储到单独的内存缓冲区中，再由单独一台机器上运行的 Windows 服务来控制这个缓冲区。状态服务全称是“ASP. NET State Service ”(aspnet_state. exe)，它由 Web. config 文件中的 stateConnectionString 属性来配置。该属性指定了服务所在的服务器，以及要监视的端口，代码如下：



```
<sessionState mode="StateServer"
    stateConnectionString="tcpip=myserver:42424"
    cookieless="false" timeout="20" />
```

在这个例子中，状态服务在一台名为 myserver 的机器的 42424 端口(默认端口)运行。要在服务器上改变端口，可编辑 HKLM\SYSTEM \CurrentControlSet\Services\aspnet_state 注册表项中的 Port 值。显然，使用状态服务的优点在于进程隔离，并可在 Web farm 中共享。使用这种模式，会话状态的存储将不依赖于 iis 进程的失败或者重启，然而，一旦状态服务中止，所有会话数据都会丢失。换言之，状态服务不像 SQL Server 那样能持久存储数据；它只是将数据存储在内存中。

4. SQL Server

ASP. NET 还允许将会话数据存储到一个数据库服务器中，方法是将 mode 属性变成 sqlServer。在这种情况下，ASP. NET 尝试将会话数据存储到由 sqlConnectionString 属性(其中包含数据源以及登录服务器所需的安全凭证)指定的 SQL Server 中。为了用恰当的数据库对象来配置 SQL Server，管理员还需要创建 ASPState 数据库，方法是运行 WinDir\Microsoft. Net\Framework\Version 文件夹中的 InstallState. sql 脚本(WinDir 是服务器的 Windows 文件夹，而 Version 是用户使用的. NET 框架版本的安装文件夹)。要配置 SQL 服务器，可以在命令行中运行 SQL Server 提供的命令行工具 osql. exe：

```
osql -S [ server name] -U [user] -P [password] <InstallSqlState.sql
```

例如：

```
osql -S(local)\NetSDK -U sa -P "" -i InstallSqlState.sql
```

在这里用户名必须是 SQL 服务器上的 sa 账号，或者具有同等权限的其他账号。有兴趣的读者可以打开这个脚本文件来了解 ASP. NET 是如何和 SQL Server 配合实现状态管理的。

卸载这些表和存储过程，可以使用 UninstallSqlState. sql 脚本，使用方法与上面类似。做好必要的数据库准备工作后，将 Web. config 文件中的 sessionstate 元素的 mode 改为 "sqlserver"，并且指定 SQL 连接字符串。具体代码如下：

```
mode="sqlserver"
sqlConnectionString="data source=127.0.0.1; userid=sa; password="
```

配置好 SQL Server 后，应用程序代码运行时就和 InProc 模式没有什么区别。但要注意的是，由于数据不存储在本地内存，所以存储会话状态的对象需要进行序列化和反序列化，以便通过网络传给数据库服务器，以及从数据库服务器传回。这当然会影响性能。通过在数据库中存储会话状态，可分别针对扩展性及可靠性来有效地平衡性能。另外，可以利用 SQL Server 的集群，使状态存储不依赖于单个的 SQL Server，这样就可以为应用程序提供极大限度的可靠性。

练　　习

1. 单项选择题

(1) 在配置文件中,标记名和属性名是________。

A. Camel 大小写形式　　B. Pascal 大小写形式

C. 都是小写形式　　D. 都是大写形式

(2) 在配置文件中,所有配置信息都驻留在________标记之间。

A. <configuration>和</configuration>

B. <authentication> 和</authorization>

C. <authentication>和</authentication>

D. <customErrors>和</customErrors>

(3) 身份验证方式不包括________验证。

A. Windows　　B. Passport　　C. Forms　　D. ASP

(4) 在 Web.config 文件中,数据库连接字符串可以有________个。

A. 1　　B. 2　　C. 3　　D. 无限制

2. 简答题

(1) 简述 Web.config 文件的主要特点和用途。

(2) 身份验证有哪几种方式?

(3) 如何使用 Web.config 文件保存数据库连接字符串? 这样做有何好处?

实　　训

实训目的

(1) 熟悉 Web.config 的配置方法。

(2) 熟练掌握数据库连接配置和身份验证配置的方法。

实训内容

在建立好的网站上新建 Web.config,实现数据库连接配置和身份验证配置。

参 考 文 献

[1] 陈建伟等. ASP 动态网站开发教程(第三版). 北京：清华大学出版社，2008

[2] 方明清等. ASP.NET 程序设计教程与实训. 北京：北京大学出版社，2007

[3] 华夏等. ASP.NET 案例实训教程. 北京：科学出版社，2009

[4] 徐祗祥等. 使用 ASP.NET 技术开发网上书店. 北京：科学技术文献出版社，2009

[5] 刘廷等. ASP.NET 开发实例完全剖析. 北京：中国电力出版社，2006

[6] 尚俊杰等. ASP.NET 程序设计. 北京：清华大学出版社，2004